Architectures for Embedded Multimodal Sensor Data Fusion Systems in the Robotics - and Airport Traffic Surveillance - Domain

Doctoral Thesis

to be awarded the degree
Doctor rerum naturalium (Dr. rer. nat.)

submitted by
Janis Schönefeld
from Hamburg, Germany,

approved by the Faculty of Mathematics/Computer Science and Mechanical Engineering,
Clausthal University of Technology,

Date of oral examination
10 June 2014

Chairperson of the Board of Examiners
Prof. Dr. Sven Hartmann

Supervising tutor
Prof. Dr.-Ing. D.P.F. Möller

Reviewer
Prof. Dr. F. Mayer-Lindenberg

Abstract

Smaller autonomous robots and embedded sensor data fusion systems often suffer from limited computational and hardware resources. Many 'Real Time' algorithms for multi modal sensor data fusion cannot be executed on such systems, at least not in real time and sometimes not at all, because of the computational and energy resources needed, resulting from the architecture of the computational hardware used in these systems. Alternative hardware architectures for generic tracking algorithms could provide a solution to overcome some of these limitations. For tracking and self localization sequential Bayesian filters, in particular particle filters, have been shown to be able to handle a range of tracking problems that could not be solved with other algorithms. But particle filters have some serious disadvantages when executed on serial computational architectures used in most systems. The potential increase in performance for particle filters is huge as many of the computational steps can be done concurrently. Again a generic hardware solution for particle filters can relieve the central processing unit from the computational load associated with the tracking task.

The general topic of this research are hardware-software architectures for multi modal sensor data fusion in embedded systems in particular tracking, with the goal to develop a high performance computational architecture for embedded applications in robotics and airport traffic surveillance domain. The primary concern of the research is therefore: The integration of domain specific concept support into hardware architectures for low level multi modal sensor data fusion, in particular embedded systems for tracking with Bayesian filters; and a distributed hardware-software tracking systems for airport traffic surveillance and control systems.

Runway Incursions are occurrences at an aerodrome involving the incorrect presence of an aircraft, vehicle, or person on the protected area of a surface designated for the landing and take-off of aircraft. The growing traffic volume kept runway incursions on the NTSB's 'Most Wanted' list for safety improvements for over a decade [82]. Recent incidents [90] show that problem is still existent. Technological responses that have been deployed in significant numbers are ASDE-X and A-SMGCS. Although these technical responses are a significant improvement and reduce the frequency of runway incursions, some runway incursion scenarios are not optimally covered by these systems, detection of runway incursion events is not as fast as desired, and they are too expensive for all but the biggest airports [113]. Local, short range sensors could be a solution to provide the necessary affordable surveillance accuracy for runway incursion prevention. In this context the following objectives shall be reached. 1) Show the feasibility of runway incursion prevention systems based on localized surveillance. 2) Develop a design for a local runway incursion alerting system. 3) Realize a prototype of the system design using the developed tracking hardware.

Contents

Chapter 1

Introduction

Contents

1.1 Motivation

Smaller autonomous robots and embedded sensor data fusion systems often suffer from limited computational power and hardware resources. Many “real time” algorithms for multi modal sensor data fusion cannot be executed on such systems, at least not in real time and sometimes not at all, because of the computational and energy resources needed for the computation of the algorithm. The amount of resources needed results from the serial architecture of the computational hardware used in these systems. Alternative hardware architectures for tracking algorithms could provide a solution to overcome some of these limitations. For tracking and self-localization sequential Bayesian filters have been applied with success to many tasks. Especially particle filters have been shown to be able to handle a range of problems that could not be solved with other algorithms. Unfortunately particle filters have some serious disadvantages when executed on the serial computational architectures used in most systems. The potential increase in performance

for particle filters is huge, as many of the required computational steps can be done concurrently. Again a generic hardware solution for particle filters can relieve the central processing unit from the computational load connected to the tracking task. The implementation in hardware results in a, with respect to the safety and reliability of the solution, more robust system, since failures of operating systems etc. do not affect the hardware based algorithms. What is true for robots is also applicable to many systems that act as agents; a selection of these application scenarios is given in the following text.

1.1.1 Application Scenarios

Application scenarios for embedded data fusion arise in every application context, where the operational constraints like reliability, size power consumption, heat dissipation, and real-time demands do not allow a software implementation. Examples like face detection in digital cameras or 3D cameras show the power of embedded data fusion solutions and their impact on everyday live. Experimental systems from the automotive field show that low level data fusion is often the crucial factor when meaningful information is required. Without a proper low level fusion adding semantics to the data is often not possible at all, while the proper fusion allows such derivations [43, 108, 89]. In the following text the applicability of the research results to important areas in the human society is explained.

Airport

Nowadays traffic surveillance at many airports is not optimal. Reasons are manifold. One of the more important reasons is that it is very difficult to integrate accurate sensors into the airport infrastructure. The sensor installation alone requires that the traffic in the area where the sensor is installed is shut down, which may include runways. More over the communication infrastructure is often not suited to transmit high volumes of data, like raw sensor data with low latency, which would require an upgrade of the communication infrastructure forcing a shutdown of at least parts of the airport and being very expensive. Embedded data fusion solutions allow the integration of sensor data fusion algorithms, like vehicle tracking into local sensors that can be plugged into existing airport infrastructure and communicate only relevant events via power line communication, a technology that exists on most airports or can be installed with little effort.
The integration of hardware sensor data fusion solutions into sensors results in a more reliable and more secure solution that can easily be integrated in planes and airport service vehicles increasing the situational awareness under unfavorable environmental conditions.

Automotive

Today's number of traffic accidents that involve cars is very high and the impact on the involved persons as well as for the society is strong. Every year ten thousands of people die in traffic accidents and more are injured. Sensor technology available today provide the basis for autonomous driving and driver assistance technology to increase traffic safety. However current data fusion solutions implemented in driver assistance technology is often limited to solutions as simple as alpha-beta filters. While these provide, given a reasonable sensor performance, a reasonable robust estimate of the current vehicle dynamics, they can hardly predict the traffic situation for more than a second and cannot reliable deal with

occlusion of targets. Particle filters have been shown to be able to reliable track targets that are occluded for some time. Therefore they provide a better situation assessment that is the basis for better autonomous driving and better driver assistance technology.

Robotics

In robotic applications it is often useful to track more than one object, e.g.: A soccer playing robots performance will increase significantly if the robot is able to track not only the ball and it's own position on the field, but also the positions of its team mates and the opponent robots. In order to track multiple objects individually, detected features must be associated with the tracked objects, discarded as false detections or associated with a track yet to be initialized. This tasks are commonly referred to as data association and track management. A pure software solution requires computational resources that, in the best case, grow linear with each track - as does the execution time. A solution using both custom hardware and software can efficiently decrease the execution time, free the CPU from the computational load and decrease the power consumption of the system.

Indoor Pedestrian tracking

Recently it has been shown [66] that it is possible to navigate indoors using a particle filter implemented on a smart phone with an accuracy of about 1.5m. The software implementation requires most of the computational resources available and drains the battery of the smart phone very quickly. Computationally similar complex tasks require between 300-900mW [8]. The prototype FPGA implementation of the particle filter discussed in this work uses about 61mW and removes the need to perform the computationally expensive processing on the central processing unit (CPU) of the smart phone. Further performance improvements can be expected by realizing the prototype as full custom hardware design. Accurate indoor navigation is the basis for a wealth of applications. While particularly the visually impaired, children, and other persons that must rely on guidance can profit from such a technology as it allows the easy navigation in complex buildings like Airports, train stations, libraries, and shopping malls the most, many task requiring indoor navigation will become much easier. In conjunction with speech-recognition and text to speech technology navigation in complex, confusing indoor environments will be much more convenient.

1.2 Research Topics

The general topic of this research are hardware-software architectures for multi modal sensor data fusion in embedded systems, in particular self-localization and tracking with the goal to develop high performance computational architectures for applications in robotics and airport traffic surveillance. The following points are the primary concern of the research:

1. Integration of domain specific concepts support into hardware architectures for low level multi modal sensor data fusion, in particular embedded systems for tracking with Bayesian filters.

2. Distributed hardware-software tracking for teams of autonomous robots in the RoboCup SPL.

3. Distributed hardware-software tracking systems for airport traffic surveillance and control systems.

1.3 State of Research

The following text provides a brief overview on the state of research with respect to the research topic of this work as defined above. The state of research is the basis for the selection of the research objectives defined in the next section. A concise description of the contribution to the state of research by this thesis is given in Section 6.1.

1.3.1 Integration of Domain Specific Concepts

The general process of multi sensor data fusion can be described by a functional model, a set of functions that can compose any data fusion system and the relation between them. The general functional model as in [75] is made of the basic functions that are necessary to perform most data fusions: Data association, data fusion, data abstraction and a knowledge representation of the current system state. One of the most used models is suggested by the Joint Directors of Laboratories (JDL).

Inferential Model for Multi-sensor Data Fusion: The inferential model provides another view on the data fusion process, and emphasizes the importance of data abstraction. Since it is not always useful to fuse information from different sources on the lowest level, data fusion and inference are often done on feature level or decision level. The revised model of the JDL distinguishes between five levels of inference from 0 up to 4, the original model had 4 levels from 1 up to 4. The definitions of the levels are [41]:

- Level 0 - Sub-object data assessment:
  Estimation and prediction of signal or object observable states on the basis of signal-level data association and characterization.

- Level 1 - Object assessment:
  Estimation and prediction of entity states on the basis of inference from observations.

- Level 2 - Situation assessment:
  Estimation and prediction of entity states on the basis of inferred relations between entities.

- Level 3 - Impact assessment:
  Estimation and prediction of effects on situations of planned or predicted/estimated actions of the participants.

- Level 4 - Process refinement:
  Adaptive data acquisition and processing to support mission objectives.

Integration of Domain Specific Concepts: Software tracking applications integrate domain specific concepts into low level data fusion mostly using interactive multiple model solutions, localized, and possibly dynamically interacting, behavior models and background knowledge like maps, combined with auxiliary information like pose estimates [93, 21, 36, 115, 47, 64]. Particular particle filters are well suited for the use of localized individual behavior models.
Works on hardware architectures that support the integration of domain specific concepts are not known to the author.

Embedded Hardware Architectures: Both functional and inferential approaches do not provide a process model which describes the canonical flow of process in detail. Because process models for hardware solutions have a strong dependency on the available technology and modeling tools, other works on this topic are not generally applicable.
Sequential Monte Carlo methods (for multi-sensor data fusion) are the topic of numerous works. Among the most popular are "Sequential Monte Carlo Methods in Practice" and "Probabilistic Robotics" [24, 111] that cover the basic principles and some advanced application specific topics. More specialized publications are available for many application areas, and in the mathematical analysis of the particle-filter algorithm and its various optimizations [19, 38, 26].
Works on hardware implementations of particle filters include [118, 7, 95, 45]. These works aim on the partitioning of the particle filter algorithm into concurrently executable parts and the optimization of these parts, as well as pipeline optimizations of the whole algorithm. The reduction of the hardware complexity of data fusion architectures, while maintaining the performance, is also topic of current research.

1.3.2 Distributed Hardware-Software Tracking Systems for Teams of Autonomous Robots in the RoboCup SPL.

In the SPL integration of domain specific concepts has been done in software tracking for example in [64, 101, 92, 107]. In the SPL hardware modifications are not allowed, making the SPL an ideal candidate to compare the performance of the hardware-software solution to the performance of state of the art software solutions. The performance of the developed hardware software solution can be compared to the performance of pure software solutions developed by other teams.

1.3.3 Distributed Hardware-Software Tracking Systems for Airport Traffic Surveillance and Control Systems.

Airport traffic surveillance is not a new topic. The growing intensity of traffic and the passenger capacity of planes led to serious accidents during the last 15 years. Some of these accidents might have been avoided if the surveillance of the airport traffic would have been better. Traditional airport surveillance was carried out by humans and radar, sometimes supported by more or less unreliable local sensors. A work published 1998 by C. Meier [75] explored the possibilities of multi-sensor data fusion using most sensors available at that time based on Kalman filters and IMM's. The work was done in cooperation with the DLR which provide an experimental test range with a complete infrastructure for testing

and verifying Advanced Surface Movement Guidance and Control Systems or parts of it on the Research Airport Braunschweig [50].
The computational resources of modern computers, recent developments in sensor technology, and sensor data fusion techniques made new approaches to the surveillance of airport traffic possible. Most of these approaches favor the use of highly reliable and accurate short range sensors. The ISMAEL [105, 106] project is an international project funded by the EU. The focus of research is the weak point in current Advanced Surface Movement Guidance and Control Systems (A-SMGCS), which is also the focus of this work. The ISMAEL project is developing magnetic field sensors that measure the magnetic field of the earth and locate disturbances caused by vehicles.
Alternative approaches use long range sensors with different modalities like sound and vision to track vehicles [88]. Often the different environmental conditions at airports make the use of certain sensors impossible. Other research directions aim at more elaborate motion and environment models and better classification of the vehicles that can be used with new sensors and commonly used sensors for a better tracking of airport traffic participants. But airport traffic surveillance is just a special case of tracking vehicles, which has been of great interest for the military since the second world war and is also of growing interest for the civilian sector. A survey is provided by Li and Jilkov in [71, 68, 69, 70, 73, 72]. Tracking any kind of objects, including self localization is essential for autonomous robots. The use of multi modal sensors in robotics did contribute to the state of the art of multi modal multi-sensor data fusion[93, 117, 112, 64, 63] . Also, faster computers have made methods of higher computational complexity available for real time tracking and multi sensor data fusion [35, 34].

1.4 Research Objectives

The following text describes the objectives of the research done on the previously described topics.

1.4.1 Objectives related to Embedded Tracking Systems

- Develop a hardware architecture for a particle filter based tracking system (HW-PFTS).
- Develop a formal distributed architecture - with a focus on concurrency - for the task of tracking multiple objects.
- Develop a formal interface for easy usage of the developed hardware.
- Develop a distributed hardware software architecture for tracking of multiple objects.

1.4.2 Objectives related to RoboCup

The robots used in the RoboCup Standard Platform League are Nao robots produced by the French company Aldebaran Robotics. Naos are small humanoid robots with limited computational resources. Although the success of teams like B-Human show that it

is possible to solve the main problems related to RoboCup, computational power still limits the performance of the robots in the RoboCup environment. As stated above self localization, tracking and image processing require the most complex operations.
Cinan and Doucet have shown that the available sensor data is a major contributing factor for the convergence of sequential Bayesian filters [19]. The number of samples used in particle filters is another major contributing factor to the precision and convergence of particle filters [111]. The computational resources of the Nao limit these important parameters for image processing, self localization and tracking. The development of a hardware solution for tracking tasks that acts as a sensor data fusion co-processor can free the central processing units from much of the computational load of tracking. The following research questions are addressed in this context.

- Integrate the HWPPTS into the ball tracking of the RoboCup Software and demonstrate the gain in performance.

1.4.3 Objectives related to Airport Traffic Surveillance

Runway Incursions are occurrences at an aerodrome, involving the incorrect presence of an aircraft, vehicle, or person on the protected area of a surface designated for the landing and take-off of another aircraft. The growing traffic volume kept runway incursions on the National Transportation Safety Board's (NTSB)'s "Most Wanted" list for safety improvements for over a decade [82]. In the past, runway incursions have led to accidents with significant loss of life. The worst accident following a runway incursion was at Tenerife, Canary Islands, Spain in 1997 were two 747 collided. Recent incidents [81] show that the problem is still existent. Although the number of runway incursions that result in an accident is small, the number of runway incursions does not decline. Statistics and results from simulation studies strongly indicate that the increase of runway incursions grows much more rapidly than the traffic volume. Depending on the airport topography an increase of 20% traffic volume may result in 140% increase of runway incursion potential for a single runway [52].
Technological responses that have been deployed in significant numbers are the Airport Surface Detection Equipment Level X (ASDE-X) and the Advanced Surface Movement Guidance and Controls System (A-SMGCS). These systems rely strongly on sensors that monitor the whole airport area, usually Surface Movement Radar (SMR) as primary sensor and Multilateration (MLAT) and Automatic Dependent Surveillance - Broadcast (ADS-B) as secondary sensors. The sensors are used as a basis for the assessment of the traffic situation and detect runway incursions. A few installations have adopted the runway lighting system and additional lights on taxiways to the traffic situation while the majority relays on Air Traffic Control (ATC) and Ground Movement Control (GMC) to transmit runway incursion alerts to the involved traffic participants. Another approach requires a retrofit of the cockpit as well as a data link between airport and aircraft [119]. Although these technical responses are a significant improvement and reduce the frequency of runway incursions, there is a big potential for further improvements. Some runway incursion scenarios are not optimally covered by these systems and the detection of runway incursion events is not as fast as desired. Non cooperative targets, which account for a large fraction of runway incursions involving commercial airliners [84, 83], are only detected by the SMR that typically has an update rate of 1 Hz and a 90%

probability of detection [18]. The cost of app. $15 million are a problem for smaller airports. Both ASDE-X and A-SMGCS are too expensive for all but the biggest airports [113]. Currently only 35 major airports in the USA field such systems [58]. Only a fraction of these use runway lighting to provide information to flight crews [84], a feature strongly recommended by several studies [119, 102]. Local, short range sensors might be a solution to provide the necessary surveillance accuracy for runway incursion prevention. In this context the following objectives shall be reached.

- Show the feasibility of runway incursion prevention systems based on localized surveillance.
- Show the superior performance of a runway incursion system based on localized surveillance.
- Develop a design for a local runway incursion alerting system based on local sensors.
- Realize a prototype of the system design using the HWPPTS.

1.5 Structure of this Thesis

The topic of this thesis is double-edged. On one side there is a very formal part dealing with the theory of embedded systems, their formal design, the theory of multi sensor data fusion, and the underlying mathematical models. On the other side is a very practical part dealing with the development of an airport ground traffic surveillance and control system. The connection of these parts is the application of the embedded system design theory on the general data fusion problem of tracking multiple targets to develop an embedded tracking system suitable for a broad range of embedded tracking applications. Thus the remainder of this work is organized as follows.
The second chapter introduces the theory of embedded systems and the theory of multi sensor data fusion in general on a higher abstraction level. A design paradigm, appropriate for the development of an embedded multi sensor data fusion system, is chosen and the formalisms that are used in the remaining chapters are explained.
In the third chapter the design paradigm - chosen in the second chapter - is applied the theory of data fusion to formally model the tracking problem. Using the Hardware-Software-CoDesign formalism the tracking problem is divided into blocks of a formal functional model. Each part of the model is analyzed, with respect to the application context of the tracking systems, to find a suitable algorithmic solution or mathematical model, from the data fusion research field. In order to achieve an optimal solution the time complexity for the execution of the model is estimated, with respect to a software solution, a hardware solution, and a hardware software solution. The result of the estimation favor a hardware-software solution where the functionality is distributed to hardware and software components, and a general architecture concept is proposed. The remainder of the chapter discusses the necessary hardware-software interfaces, and the means to implement the support of domain specific concepts.
Consequently in the fourth chapter the architecture of the tracking hardware needed for the systems architecture is developed. The chapter provides solutions for problems that arise from the violation of the mathematical models of the tracking algorithm, and

proposes an extension of the multivariate normal probability density function calculation that allows the realization of the calculation for measurements featuring arbitrary subsets of the tracked objects state space. This function provides the necessary flexibility for the mathematically sound integration of sensor measurements from a broad range of sensors, with a minimum of chip area. The chapter concludes with a discussion of the capabilities of the developed hardware based on the performance of the prototype implementation.
Chapter 5 puts the system into a real world application scenario the detection of runway incursions, a problem that is considered important enough to be of major concern for the National Transportation Safety Board of the United States of America. Because a certain amount of background knowledge is helpful to understand the environment where the tracking system is embedded, and the context of the application, the chapter starts with a survey on runway incursions, and the airport environment. In order to emphasize the advantages of a runway incursion prevention and alerting system, made possible through the use of the embedded architecture developed in Chapter 3 and Chapter 4 as a core component, the chapter continues with the theory of runway incursion prevention technology, and provides an overview of existing technological approaches to the problem. Afterward a performance comparison based on a simulation study is done, which shows the requirements for systems that can handle runway incursion incidents intractable by currently deployed solutions.
The chapter continues with a safety analysis of the developed runway incursion prevention and alerting system, which is based on extensions of the formal models used in Hardware-Software-CoDesign. Next the details of the systems design are explained by two sample installations in a simulated environment that show that the design is feasible, and a prototype realization of the design is introduced.
The last chapter concludes with a summary of the status of the research objectives, and the contributions that have been made to the state of research. Directions for future work are also discussed.
The work also contains two appendices, one on the details of a mathematical function important for the data fusion process, the other is a survey of standard key technologies used by runway incursion alerting and prevention systems.

Chapter 2

Theory of Embedded Multi Sensor Data Fusion

Contents

This chapter introduces the theory of embedded sensor data fusion. The theory of embedded multi sensor data fusion is largely concerned with applying the theory of embedded systems to the theory of multi sensor data fusion. Therefore this chapter starts with an introduction to multi sensor data fusion, continues with the theory of the design of embedded systems and the implications on this work, and in conclusion introduces the chosen design paradigm.

2.1 Multi Sensor Data Fusion

Multi sensor data fusion theory deals with the problem of getting the best information by combining multiple data from multiple sources over a specific time. The theory is primary concerned with mathematical models and formal architectures suitable to solve these kind of problems, and less with the actual means of implementing these models or architectures.

2.1.1 Definition

Multi sensor data fusion is a special case of general data fusion, although multi sensor data fusion is often referred to as data fusion. Several definitions of multi sensor data fusion are established. Mitchell [76] defines multi sensor data fusion as "the theory, techniques and tools which are used for combining sensor data, or data derived from sensory data, into a common representational format" and states "The general concept of multi-sensor data fusion is analogous to the manner in which humans and animals use a combination of multiple senses, experience and the ability to reason to improve their chances of survival." Waltz [116] says "This field of technology has been appropriately termed data fusion because the objective of its processes is to combine elements of raw data from different sources into a single set of meaningful information that is of greater benefit than the sum of its contributing parts". The comparison of data fusion with biology has also been done by Hall[40]: "Data fusion is analogous to the ongoing cognitive process used by humans to integrate data continually from their senses to make inferences about the external world." These definitions agree that data fusion is the process of integrating information from different sources into a sound representation that holds more information than the sum of the single data of the sources. This means that the goal of the process is to achieve a higher quality of information than the information contained in the single data (source). It is also well supported that the concept is very similar to that of biological organisms that combine millions of different sensing cells with different modalities to derive a high quality representation of the world.
Multi sensor data fusion may be applied to data from different sensors that might be using different modalities, as well as to data from one sensor at different times, or any combination of those.

2.1.2 Synergic Effect

The primary benefit of multi sensor data fusion, the improvement of the quality of information in a synergic process, is also the main motivation for the use of it. A synergic process is a process in which the resulting whole is more than the sum of its parts. Because a fusion of data does not automatically mean a synergy, the following points specify how multi sensor data fusion enhances the available data to achieve the synergic effect. According to Mitchell [76] multi sensor data fusion enhances the information available to a system using sensors in different ways.

1. Representation.
   The information obtained during, or at the end, of the fusion process has an abstract level, or a granularity, higher than each input data set.The new abstract level or the new granularity provides richer semantic on the data than each initial source of information.

2. Certainty.
   If V is the sensor data before fusion and $p(V)$ is the a prior probability of the data before fusion, then the gain in certainty is the growth in $p(V)$ after fusion. If V_F denotes data after fusion, then we expect $p(V_F) > p(V)$.

3. Accuracy.
   The standard deviation on the data after the fusion process is smaller than the standard deviation provided directly by the sources. If data is noisy or erroneous, the fusion process tries to eliminate noise and errors. In general, the gain in accuracy and the gain in certainty are correlated.

4. Completeness.
   Bringing new information to the current knowledge on an environment allows a more complete view on this environment. In general, if the information is redundant and concordant, there could also be a gain in accuracy.

2.1.3 Functional Model

The general process of multi sensor data fusion can be described by a functional model defined as: 'A set of functions that can compromise any data fusion system and the relation between them'. A functional model is not a process model, which would describe the canonical flow of a process in detail. Figure 2.1 shows the general functional model as suggested in [75]. This model is made of the basic functions that are necessary to perform most data fusions: Data association, data fusion, data abstraction, and a knowledge representation of the current system state. One of the most used models suggested by the Joint Directors of Laboratories (JDL) is shown in Figure 2.2.

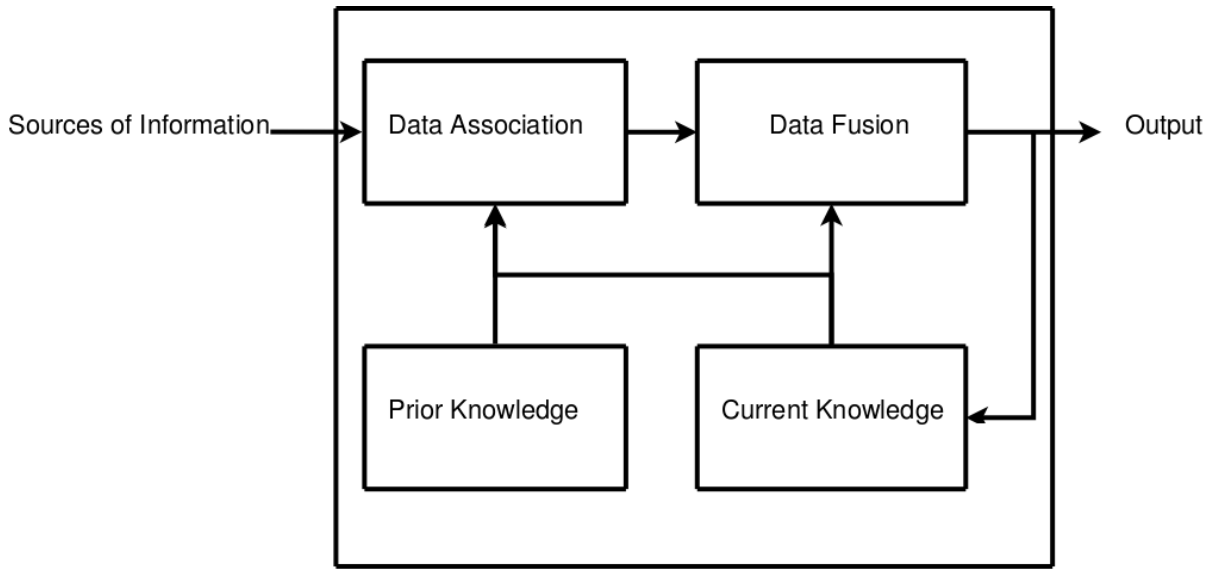


Figure 2.1: Functional model of a multi sensor data fusion module.

2.1.4 Inferential Model

The inferential model provides another view on the data fusion process and emphasizes the importance of data abstraction. Since it is not always useful to fuse information from different sources on the level of signals, data fusion and inference are often done on feature level or decision level. The levels of inference can be seen in the original functional model

of the JDL as shown in Figure 2.2. The JDL distinguishes between five levels of inference from 0 up to 4[1]. The definitions of the levels are[41]:

- Level 0 - Sub-object data assessment
  Estimation and prediction of signal or object observable states on the basis of signal-level data association and characterization.
- Level 1 - Object assessment
  Estimation and prediction of entity states on the basis of inference from observations.
- Level 2 - Situation assessment
  Estimation and prediction of entity states on the basis of inferred relations between entities.
- Level 3 - Impact assessment
  Estimation and prediction of effects on situations of planned or predicted/estimated actions of the participants.
- Level 4 - Process refinement
  Adaptive data acquisition and processing to support mission objectives.

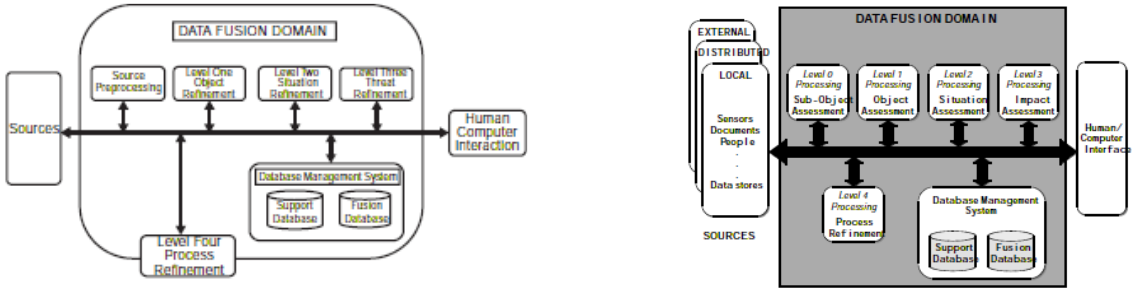


Figure 2.2: The JDL model, and the revised JDL model [41]

The levels of the JDL model touched by this work are 0, 1 ,2 and 4. Level 4 is included since the setup of the sensors is also relevant to this work. Notice that in the contextual hierarchy level 4 is located between level 3 and 2 [41].

2.1.5 System and Process Design

Biological multi sensor data fusion procedures are the result of millions of years of try and error and the resulting capability of the organism to sense its external and internal system state is amazing. Finding the right procedure for multi sensor data fusion in the context of the application is a big challenge for computer scientist and engineers. Several authors find that it is useful for this task to divide a multi sensor system in three parts. A physical domain, an information domain and a cognitive domain. Once divided, the flow of data between these parts is determined, see Figure 2.3.
When the formal processes and the formal data flow have been designed, the next challenge is to realize the system according to the constraints of the application scenario. This topic is addressed in the next section, Section 2.2.

[1]This is true for the revised version of the model, the original model has 4 levels from 1 up to 4.

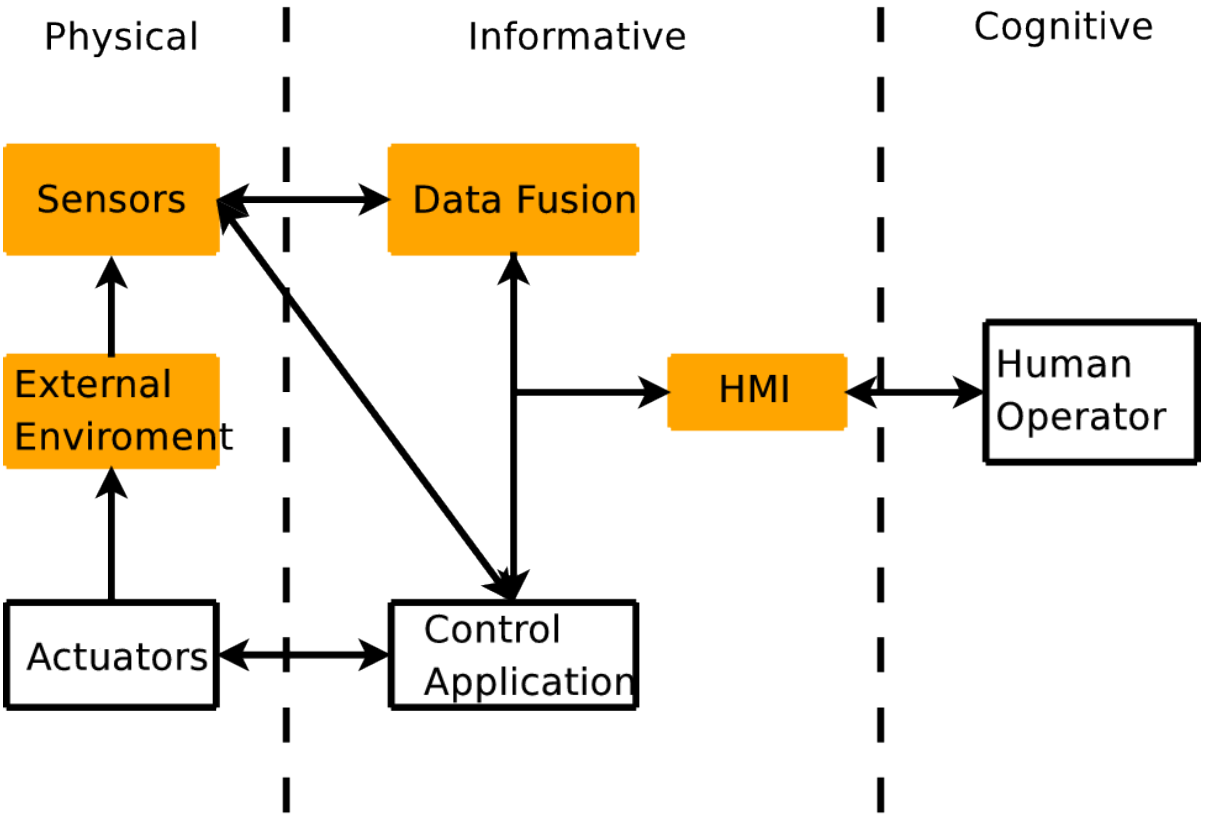


Figure 2.3: Division of the multi sensor data fusion system in three domains, the parts important for this work are highlighted in orange.

2.2 Embedding Data Fusion

The theory of advanced data fusion algorithms in general is, with respect to computer science, mostly about formal architectures, mathematical models, and the implementation of these in software. The primary motivation is that this is often the most convenient solution to implement a mathematical model, and provide a proof of concept prototype as soon as possible. Interestingly many of these algorithmic implementations are misleadingly labeled "real time" in publications. While most of these algorithm can be executed on a high performance desktop or multiprocessor system, under soft real time constraints, most reasonable application scenarios imply constraints that cannot be satisfied by the software implementation. The feasibility of efficient hardware implementations is usually not researched, but rather assumed to be given. The more complex algorithms are seldom implemented in hardware, although there are exceptions that are often very successful, like smile detection in cameras or the Asus xtion 3D sensor or multimedia extensions in modern general purpose processors (GPs). The software based approach and alternative approaches for embedding data fusion systems are discussed in the following subsection.

2.2.1 Implementation Approaches

Figure 2.4 shows a, in computer sciences common, view on the typical embedded system. The view distinguishes between the analog domain and the digital domain, where the

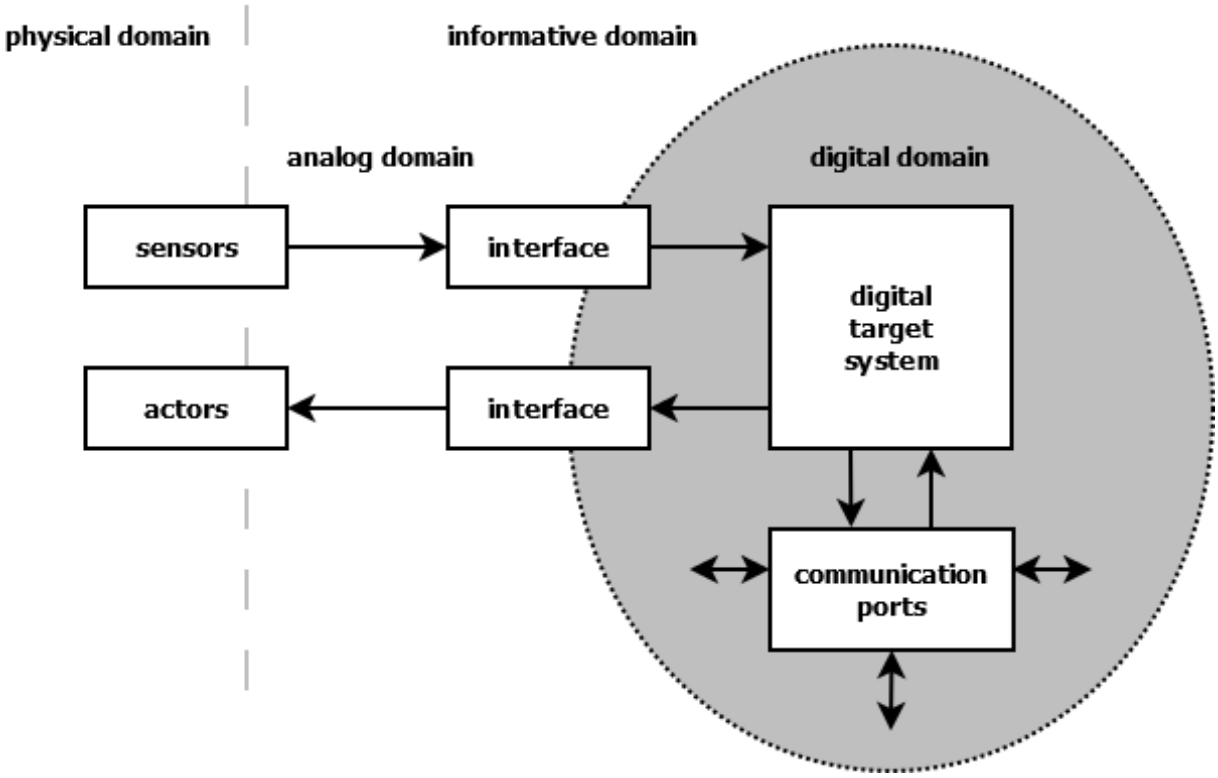


Figure 2.4: A typical embedded system.

focus is on the digital domain. While it is, in some limited cases, possible to implement data fusion systems without the usage of digital components, the majority of data fusion algorithms requires digital resources. To realize such a system 3 distinct common approaches can be identified: The software based approach, the hardware based approach, and the hardware-software based approach.

Software Based: The software implementation is one polar point of realizing the typical embedded system, as shown in Figure 2.4. A software based solution resorts to readily available of the shelf components to interface sensors and actors as needed, it has the option to resort to a great number of software libraries and programs to preprocess and post process data, allowing the scientist to focus on his research. Another advantage is the possibility to use very high level programming languages, software frameworks, and scientific/mathematical IDEs like MatLab™ or LabView™ which do not require any programming experience. Visualization and debugging are also very convenient on desktop computers. A drawback of the software implementation is that the syntax and semantics of most programming languages, as well as the architecture of desktop computers, provide only a strong support for serial computations[2], and therefore approaches that imply a parallel architecture are often not researched.

Hardware Based: Opposite to the software solution is the pure hardware solution of an application-specific integrated circuit (ASIC). ASICs provide the optimal performance

[2]Support for general purpose GPU computing can be considered weak in this context.

in terms of size, execution time, power consumption, and heat dissipation. The drawback is that their development is very expensive, and the production is also very expensive unless a great amount of ASICs is produced.

Hardware-Software Based: Between the software and the hardware solution a range of possible target technologies and design styles must be considered as well as any combination of hardware and software. See Figure 2.5 for a view on the hierarchy of target technologies, and Figure 2.6 for the different design styles. Beside GPs and ASICs, micro controllers, digital signal processors (DSPs), application-specific instruction-set processors (ASIPs), and programmable hardware, particularly field programmable gate arrays (FPGAs), are possible target technologies as well as any combination of these.

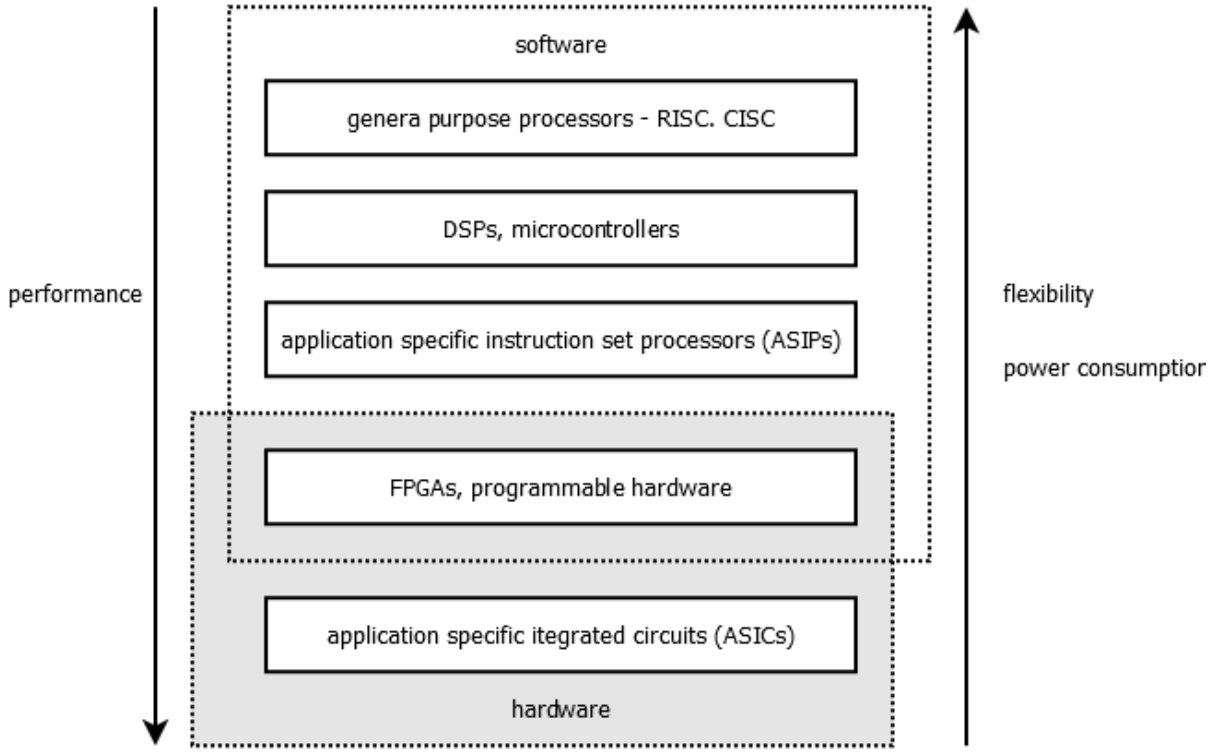


Figure 2.5: Target technologies for embedded systems.

For rapid prototyping, systems on a chip (SOCs) are available that include a GP, an FPGA, Memory, and some peripheral interfaces. With these SOCs come IDEs and operating systems (OSs). The peripherals and the GP support the convenient assessment of interfaces to connect the SOC to other resources like sensors, actors or desktop computers. They provide the flexibility of a software solution, while the FPGA allows the quick and convenient realization of algorithms in hardware. FPGA and GP communicate via own bus systems or shared memory. Alternatively soft core GP's are available for most FPGAs. The soft core provides the same functionality as the SOC GP, but it is usually less performant.It requires a part of the FPGA resources, but the interfaces between the soft core GP and other hardware parts can be freely defined, and peripherals can be custom implemented or chosen from available packages. The benefit of this approach is that one can profit from the advantages of both worlds: The flexibility and convenience of the software solution, and the performance of the hardware solution.

Design Styles: If one needs to realize a unique integrated circuit a suitable design style must be chosen. The general distinction is between semi custom design and full custom design. Full custom designs allow the optimization on all levels of the design resulting in a circuit of optimal performance, but require large amounts of time to develop and are very expensive. Semi custom designs realize the circuits through the configuration of reconfigurable components. The reconfiguration can be specified algorithmically by high level languages, therefore these devices are commonly called programmable logic devices (PLDs). Because they are produced and sold in big numbers, and programmed by the customer, PLDs are inexpensive but provide the necessary performance for most applications, while satisfying most constraints on size and energy consumption.

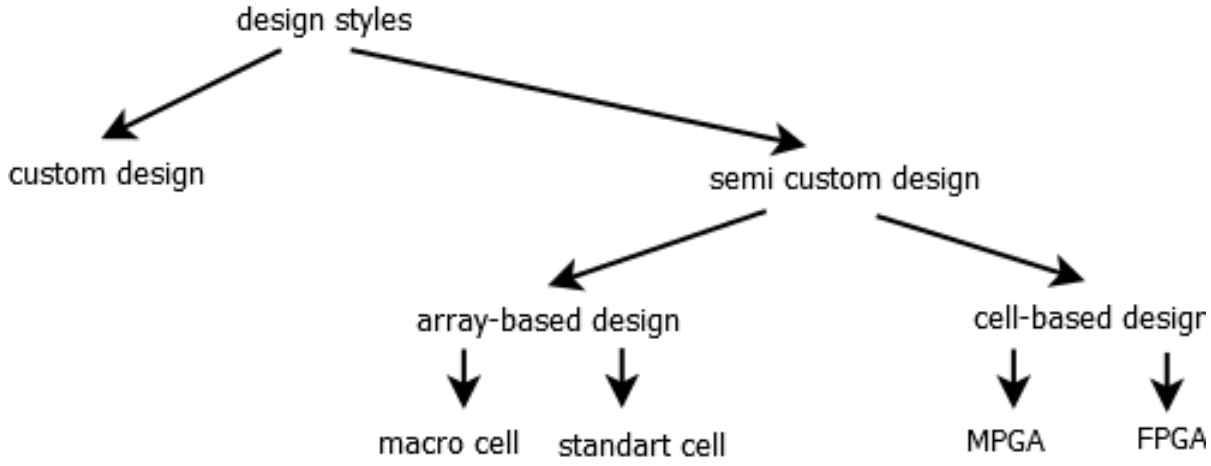


Figure 2.6: Design styles for integrated circuits.

2.2.2 Hardware Software Co Design (HSCD)

A well researched systematic approach to develop an embedded system based on a formal model is the Hardware/Software Co-Design (HSCD) approach, which focuses on the concurrent, coordinated, and integrated design of hardware and software components of a system. The HSCD approach supports a systematic design flow and provides formal models to support the design. In this work the HSCD approach is used to develop an embedded tracking solution.
The design process, see Figure 2.7, starts with a behavioral description of the system that is transformed into a formal system representation. Constraints that apply from the application scenario or limited hardware resources are integrated as formal representations. The formalisms used for the system representation will be discussed later in this section in detail. Although the schematics shown in Figure 2.7 are flat, the process itself is an iteratively deepening process, where the tasks of allocation, partitioning, and estimation are of major importance.

- Allocation is the task of choosing a valid set of hardware to realize the behavioral model. The selection of the hardware may be constrained by circumstances from the application scenario like environmental conditions, use cases, or safety standards. Allocation is not a trivial task; when formulated as a search problem the search

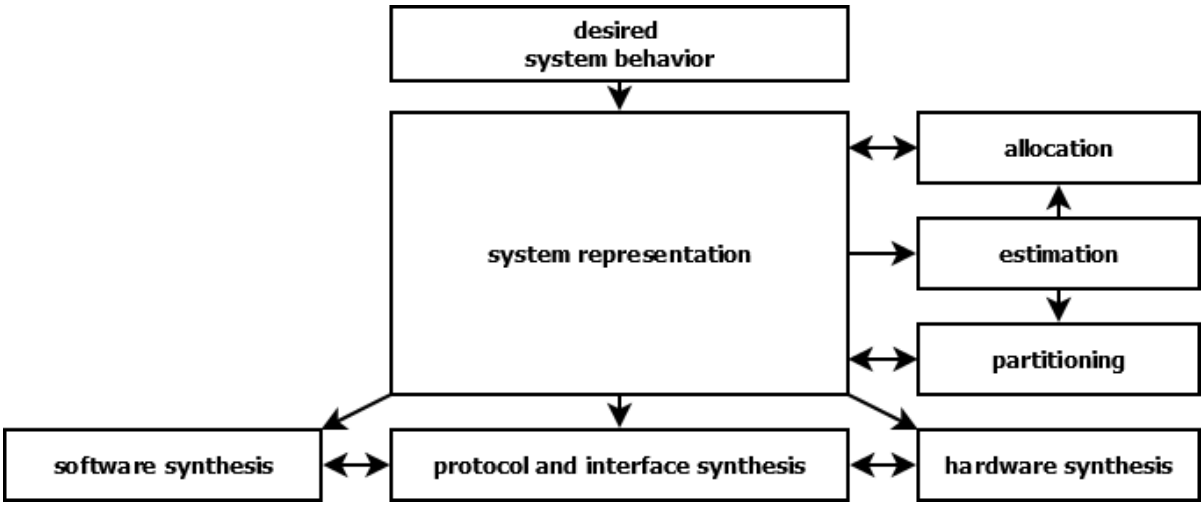


Figure 2.7: A schematic of the hardware software co design process.

space is often indefinite and nearly unlimited. During the initial design phases it is reasonable to consider abstract hardware concepts, instead of real hardware components, to reduce the size of the search space.

- During the partitioning it is determined how each part of the behavior is mapped to each part of the hardware, which also determines what is implemented in hardware and what is implemented in software. Depending on the granularity of the behavioral specification this enforces one to model the system on deeper levels to achieve optimum performance.

- During estimation the feasibility and the performance of the design is estimated. This includes the estimation of the execution times of the behavioral model as well as energy usage and other important parameters. Estimation can be done by hand on the system level, which is not an uncommon approach, using the knowledge of experts, by algorithm on the level of logic formulas and program code, or as any mixture of these two approaches.

The results of the estimation are then used to improve and refine the design in subsequent cycles of allocation, partitioning, and estimation. Once a suitable system design is found the design process continues with the synthesis. Figure 2.8 shows the process of iteratively deepening the system design, starting from the behavioral specification of the system. Further improvements of the design can be achieved if partitioning and allocation are not done on system level but on the level of subsystems.
Formal system representations that can be used for both synthesis and design are the problem graph, the architecture graph, and the specification graph. These formalisms are briefly introduced in the following text, and they are used in the following chapters of the thesis to visually demonstrate features of the system's architecture.

Problem Graph: The problem graph $G_p(V_{pc}, V_{pf}, E_p)$ is a directed acyclic graph of communication nodes V_{pc}, functional nodes V_{pf}, and directed edges $E_p \subseteq V_{pc} \times V_{pf} \cup V_{pf} \times V_{pc}$ that connect the nodes. The problem graph can be derived from the most basic

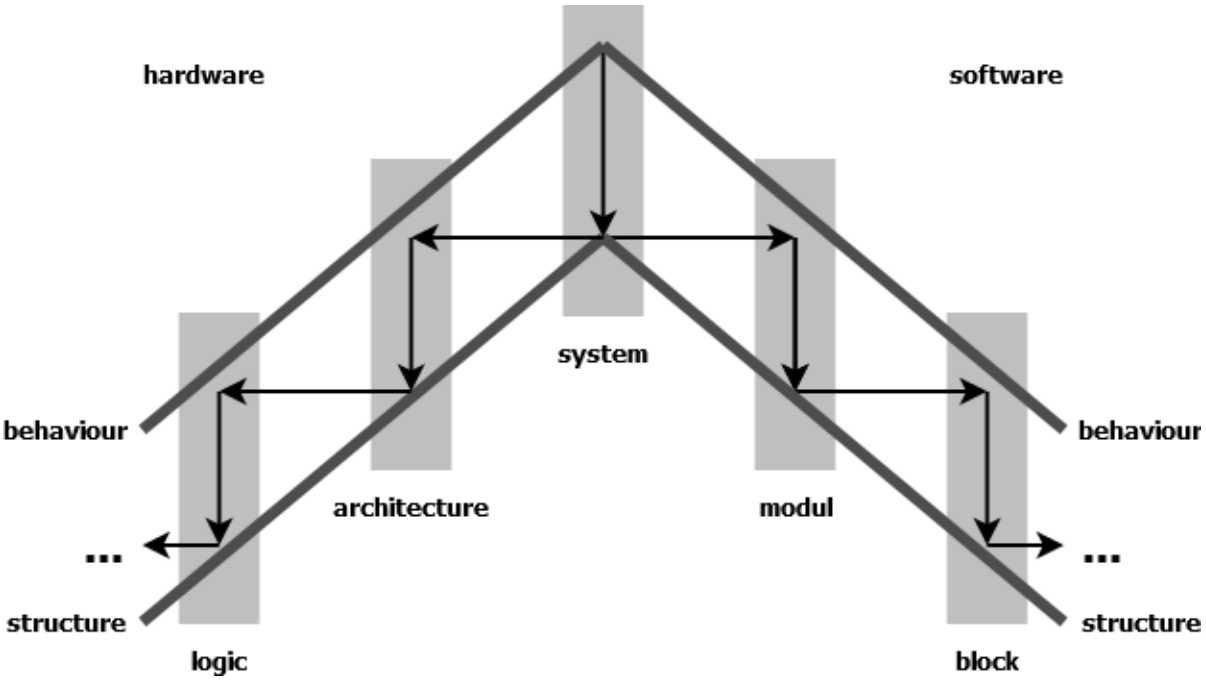


Figure 2.8: The design of an embedded system as an iterative deepening model where behavior and structure depend on each other.

form of the data flow graph (DFG), by the insertion of communication nodes between operations. Figure 2.9 shows the problem graph for a tracking problem.

Architecture Graph: The architecture graph $G_A(V_{ac}, V_{af}, E_a)$ represents the (abstract) hardware architecture of the system. It is a directed, often cyclic, graph of functional nodes V_{af} that represent computational resources like memory, ALUs, or, on system level, a complete computer, and communication nodes that represent means of communication like a bus or a network, and edges $E_a \subseteq V_{ac} \times V_{af} \cup V_{af} \times V_{ac}$ that connect the nodes. See Figure 2.9 for the architecture-graph of a tracking system.

Specification Graph: A specification graph $G_S(V_s, E_s)$ represents all possible mappings from the problem graph to an architecture graph. The specification graph includes both the complete problem graph and the complete architecture graph, as well as all possible edges from functional nodes of the problem graph to functional nodes of the

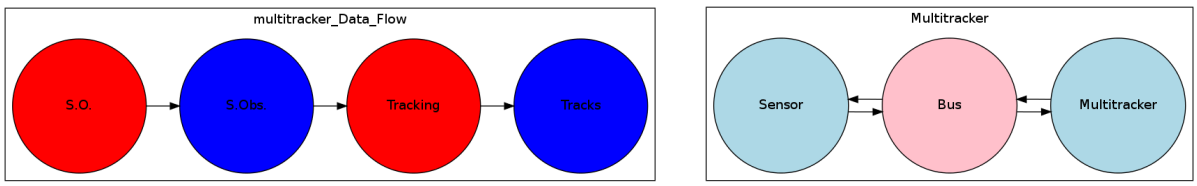


Figure 2.9: Left: Problem graph for a tracking task. Red nodes are functional nodes, blue nodes are communication nodes. Right: Architecture graph of a tracking system. Magenta nodes are communication nodes, blue nodes are functional nodes.

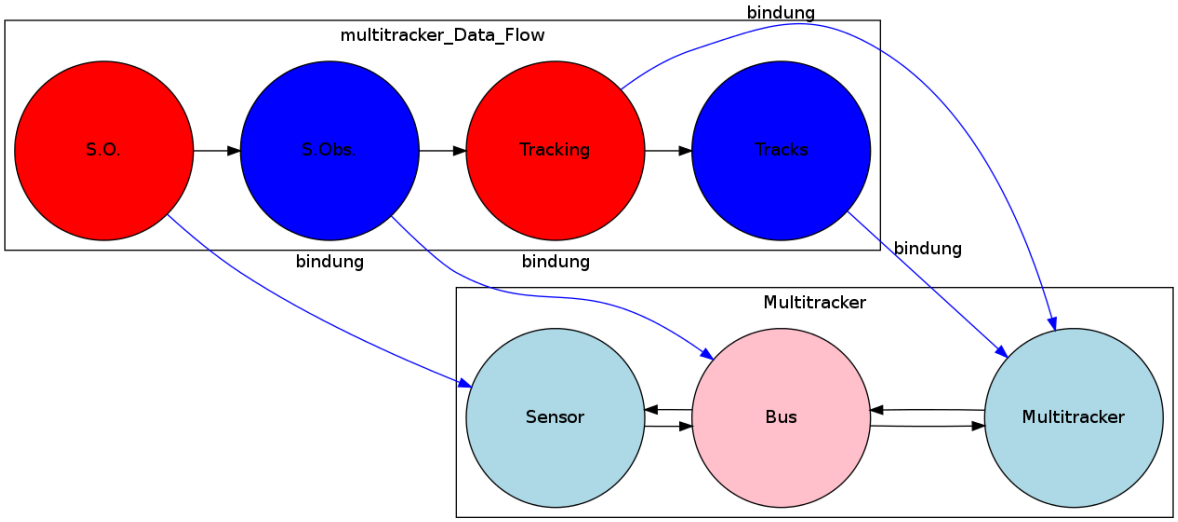


Figure 2.10: Binding of the tracking system design. Sensor observations (S.Obs.) are generated (S.O.) by the sensor and communicated over the bus to the multi tracker, where the tracking algorithm is executed and the tracks are stored.

architecture graph and all possible edges from the communication nodes of the problem graph to all nodes of the architecture graph; formally $V_s = V_{ac} \cup V_{af} \cup V_{pc} \cup V_{pf}$ and $E_s = E_p \cup E_a \cup V_{pc} \times V_{ac} \cup V_{pc} \times V_{af} \cup V_{pf} \times V_{af}$.
A binding is any subset of the specification graph. A valid binding is a binding that allows the execution of the problem graph on the hardware, which results in a number of formal constraints that apply to the binding. For example consecutive nodes in the problem graph must be mapped to neighboring nodes in the architecture graph, and functional nodes in the problem graph cannot be mapped to communication nodes in the architecture graph. Figure 2.10 shows a valid binding for the problem graph and the architecture graph shown in Figure 2.9.

Graph Refinement: Both the problem graph and the architecture graph can be refined or coarsened. During refinement a node is expanded to a new graph of the same type, with the requirement that the nodes in the boundary of the new graph have the same type as the node that has been expanded. New edges must specify the connection with the boundary of the new graph. Coarsening encapsulates a subgraph of a graph into a new node, with the requirement that the nodes in the boundary of the subgraph have the same type, which determines the type of the encapsulating node. Edges from the boundary must be merged to new edges that connect the encapsulating node with the graph. Figure 2.11 shows the refinement of the architecture graph shown in Figure 2.9.
Bindings and specification graphs change accordingly to the coarsening or the refinement of the graphs. New edges must be included and existing edges merged, to reflect the changes in the architecture graph and the problem graph. On higher abstraction levels valid bindings include bindings that map nodes of the problem graph to multiple nodes of the architecture graph, to represent distributed execution and storage, under reasonable constraints. Figure 2.12 shows a binding for a refined problem graph and a refined architecture graph. The binding shown is explained in detail in later chapters and, in this

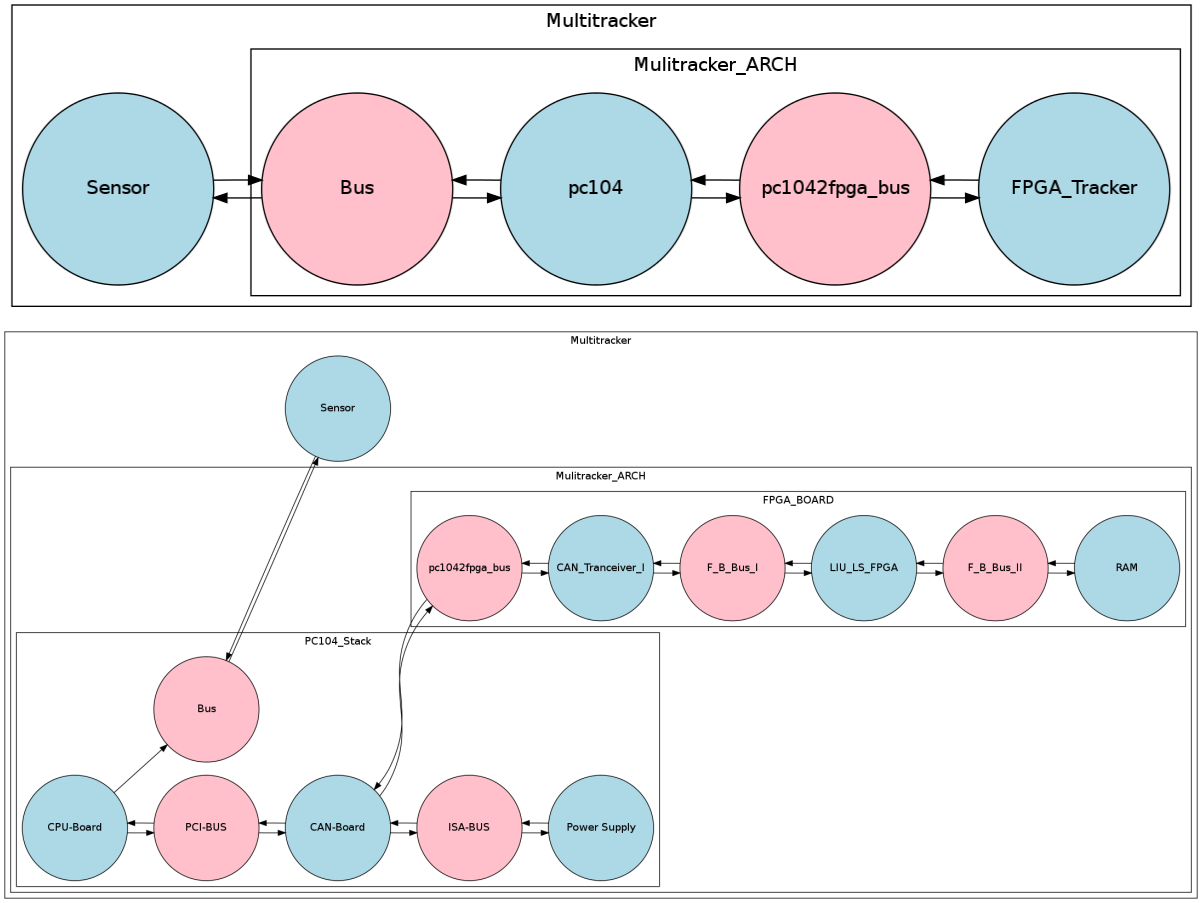


Figure 2.11: Refinement of the architecture shown in Figure 2.9. In the upper image the "Multitracker"-node is refined to a pc104 connected to a FPGA. In the lower image the pc104-node and the FPGA-node are refined to more detailed architectures.

section, merely demonstrates the effect of coarsening and refinement on the binding.
This work follows the hardware software co-design approach to support the design and development of the architectures for the embedded multi sensor data fusion algorithms and the descriptive formalisms to describe the embedded system design and its components. Because of the application context the formalism was extended to support the semiautomatic estimation of system characteristics related to functional safety, the details on this are discussed in Section 5.5. In the following text the formalism is used for the following purposes:

- Navigation: The visualization of the graphs allow the convenient orientation during the, unavoidable, ex-courses into mathematical aspects, algorithmic details etc., with respect to the data fusion process and the system architecture. Graphs where only the relevant parts are colored provide a one look orientation during text passages concerning the very details of topics only remotely relevant to the architecture itself.

- Lucidity: The visualization of the graphs allows the intuitive understandable display of the relations between different parts of the formal models, algorithms, processes,

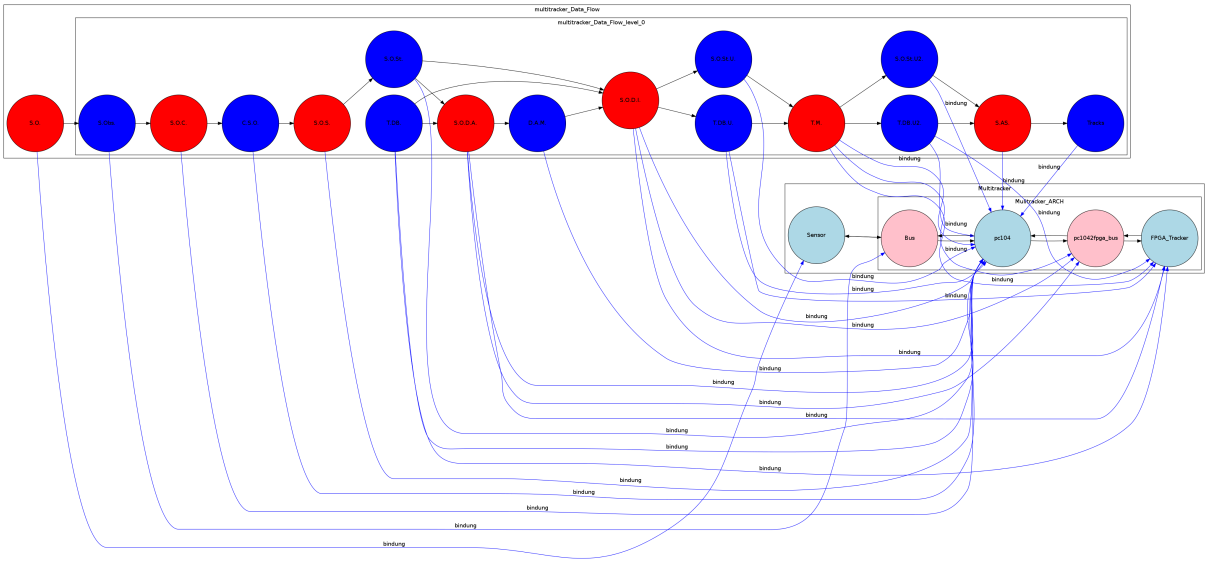


Figure 2.12: Binding on higher abstraction level; nodes in the architecture or problem graph may be mapped to multiple nodes in the other graph.

system architecture, hardware, and software. While there are exceptions, the graphs are generally better suited to provide a more convenient view on the internal relationships than a text.

- Demonstration: The easily accessible semantics of the graphs, particularly the binding, supports the convenient assessment of the reason-ability of various aspects of the systems architecture.

Figure 2.12 illustrates the above points. Even a concise description of systems internal relations will not be as easily accessible as the binding graph shown in the figure. Distributed calculations and data storage are visible at the first glance, identifiable by the multiple edges from single nodes in the problem graph to multiple nodes in the architecture graph. It is also immediately accessible that all nodes of the problem graph are bound to at least one part of the architecture graph. A closer look reveals that there is a path trough the architecture graph for every path in the problem graph, and the hierarchical relationships of both the architecture and the data fusion process are clearly visible.

2.3 Summary

This chapter introduces the theory of multi modal sensor data fusion, it's goals, standard architectures, and the general process and system design. Common approaches to realize an embedded data fusion system are explained as well as different design styles. The Hardware/Software-Co-Design approach and the relevant formalisms are introduced, and its relation to the thesis has been explained.

Chapter 3

Hardware-Software Tracking

Contents

In this chapter the difficulties and limitations that apply to the mathematical theory of tracking an object, and the impact of these on the tracking performance are discussed. The sources of uncertainty and error are identified, and they are analyzed with respect to the computational resources needed and to the tracking performance of the developed system. The embedded tracking system has been partially developed in the RollMops project, which is part of the national air transport research program "Competitive Airport (WWF)" by the Federal Ministry of Economy and Technology (BMWi) special program LuFo IV. Nevertheless the developed architecture is suitable for a wide range of tracking problems. The chapter starts with the theory of tracking against this background, followed by a discussion of the system design. The chapter concludes with an analysis of the possibility and feasibility to integrate domain specific information in the data fusion process.

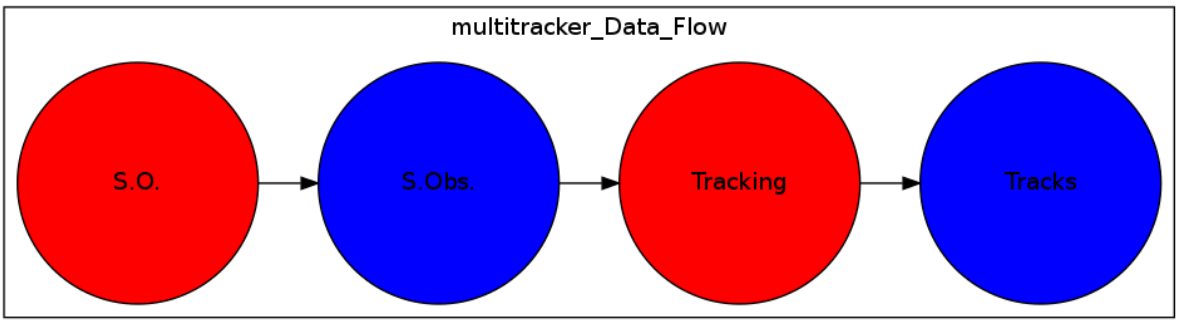


Figure 3.1: Problem graph for the tracking problem. 1 or more sensors (S.O.) generate measurements or observations (S.Obs.) from which tracks (Tracks) , that represent the state of vehicles in an area, are derived (Tracking).

3.1 Theory

This section covers the theory of tracking multiple targets in a cluttered environment, that is the mathematical models and the errors and limitations of these models, with respect to the airport ground traffic situation assessment application scenario. Most of the discussed topics apply to general 2D map tracking, which has a broad range of application scenarios. The task is formally expressed by the problem graph shown in 3.1, and it includes the following nodes:

- Sensor Operation (S.O.): A functional node representing 1 or more sensors. The sensor(s) measures it's (their) environment and generate(s) observations that describe the measurement on the abstraction level of signals or features.
- Sensor Observation(S.Obs.): A communication node representing the observations of the sensor(s)
- Tracking: A functional node representing the tracking task. Sensor observations are integrated to extract meaningful information, in the form of the position and movement of objects, from the measurements of the sensor.
- Tracks: A communication node representing the abstract and human understandable information about the position and movement of objects in the area monitored by the sensors.

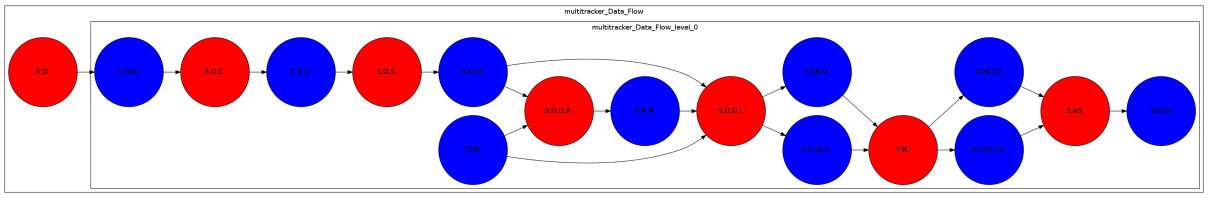


Figure 3.2: The refined problem graph of the tracking task.

The heterogeneous environment and the complexity of the tracking task require adequate mathematical models and suitable representations to allow for the realization as an embedded system. In the following text aspects of the different tasks and involved data types are discussed on the basis of the refined problem graph of the tracking task, shown in Figure 3.2. The refined problem graph is used to show how the different parts of the algorithm and the data types are related to the formal modeling of the task. The refined problem graph includes the following nodes.

- Sensor Operation (S.O.): A functional node representing one or more sensors. The sensor(s) measures it's (their) environment and generate(s) observations that describe the measurement on the abstraction level of signals or features.
- Sensor Observation(S.Obs.): A communication node representing the observations of the sensor(s)
- Sensor Observation Conversion (S.O.C): A functional node that represents the conversion of a sensor observation in a sensor specific format to a common representational format. The common representational format is useful to integrate sensor measurements and other sources of information into a consistent model to represent the distribution of the possible states of the environment.
- Converted Sensor Observation (C.S.O.): A communication node representing the sensor data in the common representational format.
- Store Observation from Sensor(S.O.S): The converted sensor data is stored in an organized way together with sensor data of earlier execution cycles, which can still be of use for the data fusion process.
- Sensor Observation Storage (S.O.St.): A data base holding the sensor observations of the present and recent past that are useful for the tracking task.
- Track Data Base (T.DB.): A distributed data base holding the information about the belief of the vehicles state variables like position, heading, etc..
- Sensor Observation Data Association (S.O.D.A.): Based on the information from the track data base and the stored sensor observations, sensor data is associated to tracks representing vehicles in order to decide which data is integrated into which track.
- Data Association Matrix (D.A.M.): A communication node that represents the data association matrix (DAM). The DAM is a data type that maps observations to their likely origin.
- Sensor Observation Data Integration (S.O.D.I.): A functional node that represents the integration of the track data, background knowledge, and sensor observations into a consistent model of the state of the world. The new estimation of the state of the environment is reflected in the update of the track data base and the sensor observation storage.

- Track Data Base Update (T.DB.U.) : The integration of the sensor data in the track data base results in the updated track data base.
- Sensor Observation Storage Update (S.O.St.U.): The integration of the association information results in an updated sensor observation storage, where observations integrated in the preceding operation have been deleted.
- Track Management (T.M.): A functional node representing the task of managing tracks. This is mostly the initialization of new tracks and the deletion of old tracks. The task results in an update of the track data base and the sensor storage, as sensor observations that could not be associated with any existing track might be associated with a new instantiated track, or tracks were not associated to any measurement for a long time are deleted.
- Track Data Base Update 2 (T.DB.U2.): The instantiation of new tracks and removal of obsolete tracks results in a second update of the track data base.
- Sensor Observation Storage Update 2 (S.O.St.U2.): Sensor observations used during track management and not needed anymore have been deleted.
- Situation Assessment (S.AS.): This functional node represents the task of assessing the environmental situation on a higher level than that of single tracks, for example the traffic situation in an environment, which also integrates knowledge about the interaction of traffic participants, schedules, or traffic rules. It's output is a human understandable list of tracks representing the tracked objects.
- Tracks: A communication node representing the abstract and human understandable information about the position and movement of objects in the area monitored by the sensors.

3.1.1 Common Representational Format

In order to integrate the multi modal data of multiple data sources it is useful to have a abstract representation that allows the expression of the belief over the state of the system that is monitored. This abstraction is commonly referred to as the common representational format (CRF). It has to be chosen in such a way that it is possible to transform, with a negligible corruption or loss of information, the data of arbitrary sensors into the CRF for optimal usage in the application context. Another necessary requirement is the computational efficiency of the chosen CRF. To localize the transformation to the CRF in the tracking process refer to Figure 3.3 that shows the refined problem graph of the tracking problem. The colored nodes deal with the transformation of the data from various sensors and tracks to the CRF.
The common representation chosen for the tracking task is a feature based 2 dimensional map with Cartesian coordinates, where objects are represented by a probability density over the position, heading and the 1 derivative of these values, as well as an id that identifies the object, and a time-stamp to determine the last update of the object. Probability densities are represented by normal distributions. Special properties like points of reference, signatures, or boundary can be accessed using the id of an object and a database but

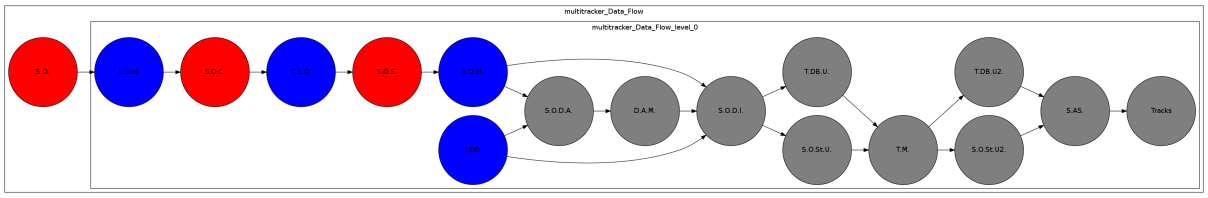


Figure 3.3: Parts of the problem graph that are primarily concerned with the transformation of data to the common representational format (CRF).

are not part of the CRF. The choice of the CRF with respect to the application scenario is motivated on the following pages.

Time-Stamp

Time-stamps are required for most tracking algorithms, to align sensor observations or target positions in a common time line. The time is given by a tuple of seconds and microseconds as commonly used to represent Greenwich mean time (GMT). GMT is also the common time axis used. However some components or sensors do not provide time-stamps, which must be inferred by the tracking system from the characteristics of the system components and the architecture. The topic of temporal alignment is addressed later in the following text.

2D Map

In many application scenarios it is useful and necessary to focus on the most relevant aspects of the world and to use abstractions such as point wise linearization, instead of modeling the world with arbitrary precision. The 2 dimensional map is an excellent example. While the worlds coordinate system is spherical, structured artificial environments and moment limitations that arise from physical constraints, allow the assumption of flat 2 dimensional maps with a isometric coordinate system, where the position of a vehicle is expressed as a triple (x, y, θ) stating the position and the heading of the vehicles own coordinate system.
While the 2 dimensional map is a nearly perfect approximation at small scales, there is a difference on bigger scales. A floor in an regular office building will be mapped with sufficient accuracy, but exact representation of an airport requires a spherical coordinate system unless the effort has been made to remove the earths curvature for a perfectly plain airport surface. Fortunately areas of the size of major airports for example cannot be mapped, using a barrel projection, to an isometric 2 dimensional space with a difference smaller than s a few meters at the boundary of the map [100]. The distortion results in a systematic error that is depending on the characteristics of the sensors and the required accuracy of the tracking, negligible for most sensors, which is shown in the discussion of the sensor value normalization later in the following text.

Normal Distribution

Because of the uncertainty of sensors and vehicle movements, reasonable accurate algorithms require the integration of these uncertainties. The normal distribution provides a

compact closed form representation that is suitable to represent most sensor measurements as well as the state of the dynamics and the position of vehicles on the map. Because of this ,and its efficient representation, it is perfectly suited for the common representational format and for the exchange of information between components of the tracking system. It is also widely accepted as an input format by many other tracking systems and human machine interfaces (HMI).

ID

With respect to the possible high data rates, the limited resources available in reasonable target technologies used to realize components of the system, and the flexibility of the implementation both in software and hardware it is reasonable to limit the CRF to the properties of the targets that are required for the tracking task and may be subject to constant change. Properties that are not subject to change, or which are just needed at special components, can be represented individually by a suitable chosen representation, and they can be referenced by the id of the CRF, e.g.: The outline of a vehicle that is used for the visualization in a HMI may be inferred by the information in a database that object with id=X is a Boeing 747.

3.1.2 Spatial Alignment

Spatial and temporal alignment or registration is a process where local coordinates and time-stamps are mapped to a common time axis and a common coordinate system defined in the CRF. Figure 3.4 shows the which nodes of the refined problem graph are concerned with the spatial and temporal alignment.

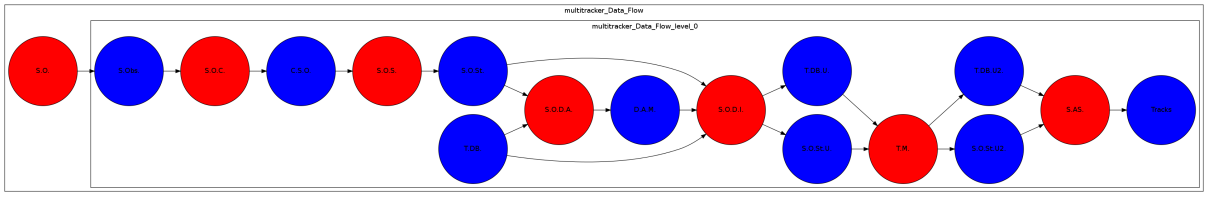


Figure 3.4: Localization of the spatial and temporal alignment in the refined problem graph.

Conversion of probability distributions from polar to Cartesian coordinates

The majority of sensors encountered in the airport, automotive, and robotics environment use their own polar coordinate system with their own measurement units. To update the belief over the targets state these values have to be passed through a function that maps the sensors measurement space to the common representational format. Almost all sensors assume a Gaussian distributed error for their measurements, where the individual errors are not correlated . Unfortunately the transformation from polar to Cartesian coordinates is not linear, thus the mapping of a linear Gaussian probability density function in polar coordinates to a linear Gaussian probability function in Cartesian coordinates - and vice versa - induces an error. Common approaches to deal with this challenge are: The

unscented transform, the working point linearization, and the inverse transformation. These are introduced in brief in the following text, and there suitability for the use in an embedded system is discussed.

Unscented Transform The unscented transform is a transformation developed to deal with non linear transformations of linear defined probability density functions. The underlying idea is that is is easier to approximate the transformed probability density than to approximate the transformation. The unscented transform has been developed by Julier and Uhlmann in 1997, see [60, 61]. A brief introduction can be found in [111]. The unscented transform aim to capture and preserve the moment of the probability density function. The algorithm, which is in depth described in the Appendix B, uses so called σ-points to capture the moment of the untransformed probability distribution and computes a normal distribution from the transformed σ-points. For most sensor data given in polar coordinates the transformation is straightforward, the σ-points, for each dimension $\mu - \sigma, \mu$ and $\mu + \sigma$, are transformed from polar coordinates to Cartesian coordinates, and the mean μ and covariance Σ is then calculated from the transformed σ-points. The error induced by this transformation is strongly dependent from the amount of uncertainty of the sensor measurement and the distance to the sensor, see Figure 3.5.

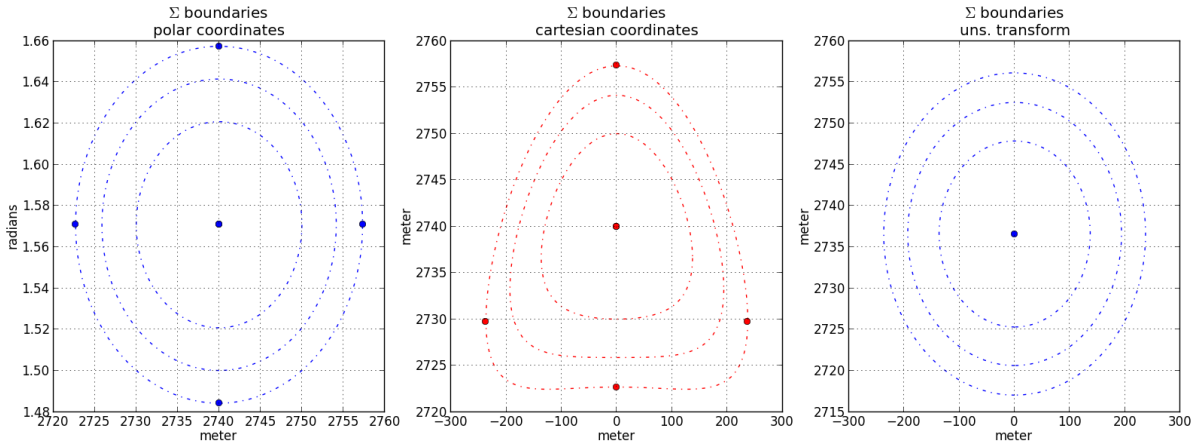


Figure 3.5: Illustration of the error induced by the unscented transform. The leftmost image shows the measurement in the sensors polar coordinate system. The rightmost image shows the approximate PDF from the unscented transform. The image in the middle shows the exact mapping of the measurement to Cartesian coordinates. In contrast to the working point linearization , which preserves the mean, the mean is about 7 meters from the direct mapping, but the moment of the PDF is preserved better than by the working point linearization. The σ-points are shown on the σ-boundary, at points $\mu - \sigma$ and $\mu + \sigma$.

Working point linearization This approach tries[1] to preserve the mean of a normal distribution by linearization the transformation function at the position of the mean. As

[1]Actually the mean of an normal distribution approximation of the real transformed PDF is most times not exactly preserved. See [111] for some vivid examples.

stated before, state transitions and measurements are in practice often not linear, or no linear transformations are required to convert the data to the CRF. A transformation from polar to Cartesian coordinates using a linear function results in a preservation of the mean, but the moment of the probability density function is lost. The working point linearization, which is also the basis for the Extended Kalman Filter (EKF), approximates the non linear transition function by a linear function tangent to the mean - the working point - of the normal distribution. Many techniques exist for the linearization of nonlinear functions. The most popular is the first order Taylor expansion. However the derivation of the tangent function used for the linearization is more complex than the unscented transform, and the performance of the unscented transform has been found to be superior to that of the working point linearization in many cases [111]. Because of the indications from the literature and the elegance of the unscented transform algorithm, which is better suited for a hardware implementation, the unscented transform has been chosen as a transformation method, and the performance of the working point linearization has not been compared experimentally.

Inverse transformation: Using a non parametric representation[2] of the belief (of the target position), like the particle filter does, it is possible to use the inverse function f^{-1} to transfer the belief representation to the sensors coordinate system, and update the individual values of the particle filter that make up the belief in the sensors coordinate system. For particle filters, where the probability density is given by the density of a finite number of samples, this method has the nice property of removing the mathematical error that is normally induced by the transformation of the sensors measurement to the CRF. The drawback of this approach is that it requires the computation of f^{-1} for every sample in the sample set of the particle filter, instead of the one time transformation of the sensor measurement itself. Because of the expected additional hardware requirements, which also would result in a higher energy usage, the unscented transform was chosen as a method to convert the probability distributions, and not the inverse transformation.

Mathematical error analysis

When applying the unscented transform to the problem of converting normal distributions from polar coordinates to Cartesian coordinates the results strongly depend on the parametrization of the normal distribution. The bigger the uncertainty, the bigger the error of the approximation. The transformation is quite precise for universal medium range radar (UMRR) measurements and Laser measurements, but has a significant error for long distance measurements of older surface movement radars (SMR). Figure 3.6 illustrates the influence of the angular uncertainty on the σ-boundaries of the transformed probability

[2]Opposed to commonly used parametric representations that use a parametrized mathematical function or model with a fixed number of parameters for the representation of objects or distributions and assume a fixed structure of the model, non parametric representations have a flexible number of parameters that depends on what one wants to represent and the structure is determined from the given data. E.g.: A normal distribution is a parametric representation where the parameters are chosen in such a way that the fixed structure assumed best fits the data at hand. A set of samples is a non parametric representation that does not assume a fixed structure of the data, but requires a method to reasonable extract the structure from the sample set.

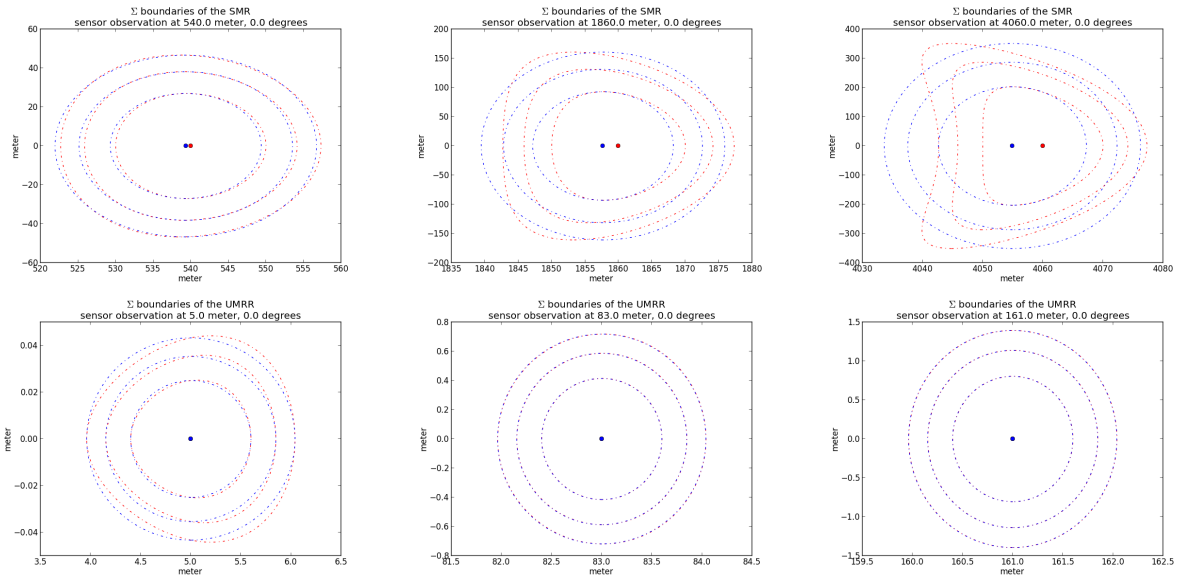


Figure 3.6: Sigma boundaries of close, medium, and long distance measurements, and the σ-boundaries of the approximation by the unscented transform for SMR (top) and UMRR (bottom) sensors. The red line plots the σ-boundaries of the transformed PDF. The blue line the approximation by the unscented transform. The dots mark the mean of the PDF.

density function. With growing distance the shape of the transformed probability density functions σ-boundaries changes from a ellipsoid shape to a triangular shape.
To determine the error induced by the transformation one should consider the following facts. The probability that a variable x is inside an interval $[a, b]$ is given by the integral under the probability density function, where x is a possible state of the system, in most cases the position of a tracked mobile, and the probability density function is given by a sensor observation z. With respect to the accuracy of a tracking system however the error of the mean and the error of the main moment may be more important than the overall error. So not only on the absolute error, the difference between the local probability density at any point of the probability density functions, but on the effect to the convergence behavior of the tracking algorithm is of major concern.
Because the normal distribution is a continuous probability density function (continuous PDF) its appliance in digital computation is not without problems. As mentioned above, a continuous PDF f allows the computation of the probability that a value x is inside an interval $[a, b]$ by integration over the interval, $P(x \in [a, b] = \int_b^a f(x)dx)$, but the probability at any point is infinitely small $P(X = x) = 0$. With a digital representation it is only possible to compute the probability density at discrete points or over intervals within the representative capabilities of the digital number representation.
Therefore the continuous probability density function has to be approximated by a discrete probability density function. Using the law of total probability this is done by sampling over the digital space, which includes all points where the carrier of every function applied to the digital representation is not 0.
There are different approaches to compute the probability given at a specific discrete point in digital space. For equidistant points one approach is to integrate the probability

density function over the interval $[x - \frac{\beta}{2}, x + \frac{\beta}{2}]$, where β is the distance between the points. Using a working point linearization assumption, this can be done very efficient using equation 3.1, where x is the position of the point (the system state) and μ and Σ are the mean and covariance of a normal distribution $z = N(\mu, \Sigma)$ representing a sensor observation.

$$p_{polar}(x|z) = \eta N(x, \mu, \Sigma)\beta^2 \tag{3.1}$$

For the following text the notation η will denote the normalizer in Bayes rule variable that ensures that the law of total probability is satisfied. The normalizer is computed given by $p(z)^{-1}$ as in Equation 3.2

$$\eta = \frac{1}{\sum_{x' \in X} N(x', \mu, \Sigma)\beta^2} \tag{3.2}$$

Here X is the set of points representing the digital representable system state (target position). Although the normal distribution is never 0 in continuous space it is sensible to restrict the set of points $x' \in X$ to the points where the magnitude of the normal function is not negligible $N(x', \mu, \Sigma) > \epsilon$. Equation 3.3 shows that the the area β^2 can be canceled out to make the computation even more efficient. This approach can be applied to higher dimensional spaces.

$$p_{polar}(x|z) = \frac{1}{\sum_{x'} N(x', \mu, \Sigma)\beta^2} N(x, \mu, \Sigma)\beta^2 = \frac{N(x, \mu, \Sigma)}{\sum_{x'} N(x', \mu, \Sigma)} \tag{3.3}$$

To determine the error induced by the transformation one has to sum up the difference between the probability in polar coordinates $p_{polar(x|z)}$ and the probability in Cartesian coordinates $p_{cartesian}(x|z^{usc})$ under the transformed probability density function z^{usc}. To calculate the probability density in Cartesian coordinates the points are converted from polar coordinates to Cartesian coordinates using the function f^{p2c} and applied to the unscented transform of the probability density function N^{usc} in Cartesian coordinates. The problem that arises from the conversion of polar coordinates to Cartesian coordinates is that the points equidistant in polar coordinates are not equidistant in Cartesian coordinates and vice versa, and the area in polar coordinates is not the same as the areas in Cartesian coordinates for each sampling point (see Figure 3.7).

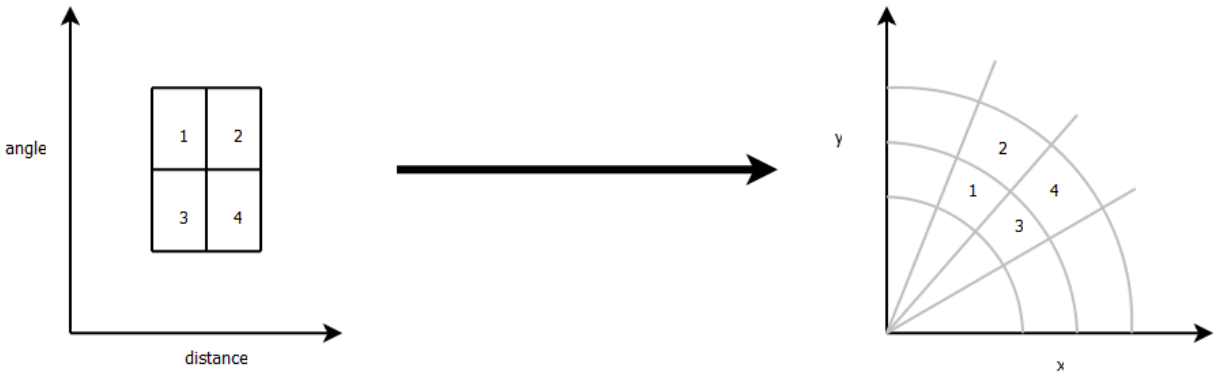


Figure 3.7: Conversion of an isometric pattern from polar to Cartesian coordinates.

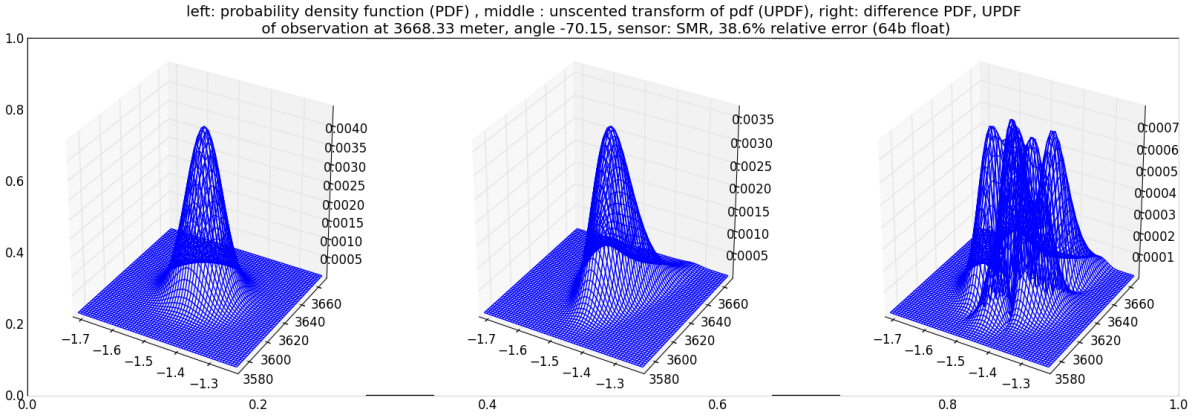


Figure 3.8: Difference between the original probability function and the unscented transform. For this plots the probability density function has been sampled with 900 data points in an isometric pattern over the interval $[\mu - 5\sigma, \mu + 5\sigma]$. The plots show a far away measurement where the difference between the normalized volumes sums up to 38.6% of the volume of the polar probability density function.

Therefore each sampling point has to be weighted by the area represented by this particular sampling point in the coordinate system the isometric pattern is transformed to. In this case Equation 3.3 can only be applied to the isometric sample point pattern. For the transformed pattern the area has to be computed for each sampling point x and used as a weight β for the sampling point, the adequate equation for the probability of a sampling point is:

$$p_{cartesian}(x|z^{usc}) = \frac{N^{usc}(f^{p2c}(x), \mu, \Sigma)\beta}{\sum_{x'} N^{usc}(f^{p2c}(x'), \mu, \Sigma)\beta'} \tag{3.4}$$

Equation 3.4 shows that all constant factors can be removed from the computation of the weight, leaving only the shortest distance to the begin and end of the area used for the calculation of the weights as variables.
To error $E_{\omega,\theta}(z^{usc})$ of the unscented transform z^{usc} of a single observation z at a specific angle θ and distance ω is then given by the the equation:

$$E_{\omega,\theta}(z^{usc}) = \eta \sum_{x} |p_{polar}(x|z) - p_{cartesian}(x|z^{usc})| \tag{3.5}$$

The magnitude of this error is strongly dependent on the characteristics of the sensor and the position of the observation's mean in sensor coordinates. An example visualization of the error analysis is shown in Figure 3.8. The average error $\overline{E}_{sensor}$ for a sensor can be obtained by sampling Equation 3.5 over the reasonable angular range Θ and distance range Ω of the sensor using the equation:

$$\overline{E}_{sensor}(Z^{usc}) = \frac{1}{\#\{\Omega \times \Theta\}} \sum_{\omega \in \Omega} \sum_{\theta \in \Theta} E_{\omega,\theta}(z^{usc}) \tag{3.6}$$

For some of the more common sensors this results in the values shown in Figure 3.9 and presented in Table 3.1. The distribution of the error with respect to the magnitude of the error is plotted in Figure 3.10. Depending on the uncertainty in angle σ^{Θ} and distance σ^{Ω}

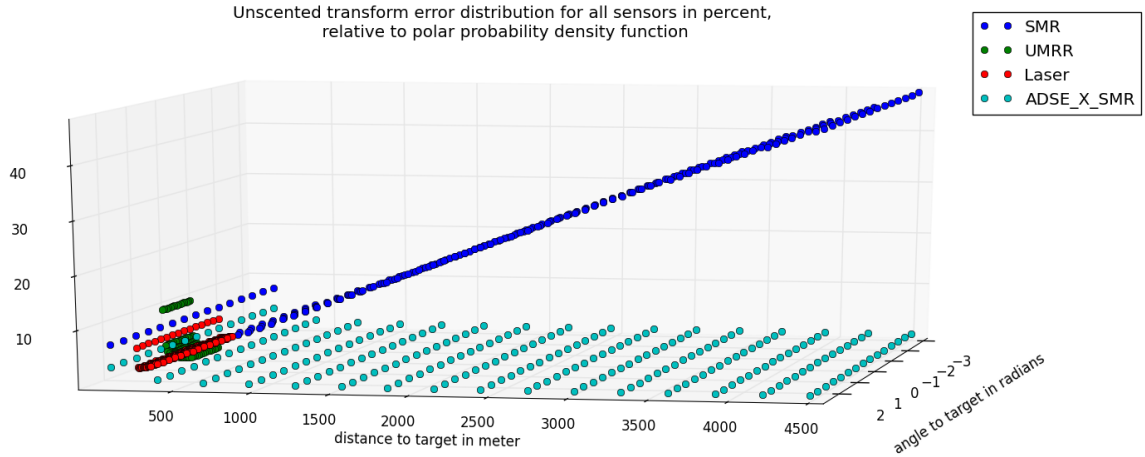


Figure 3.9: The error distribution of the unscented transform for the reasonable range and angle of four common sensors. Except for the surface movement radar (SMR) the error is very low except at very close distances. Because the absolute uncertainty of Laser and UMMR measurements are very low, the relative high increase at very close ranges does not significantly affect the performance with respect to most tracking scenarios.

the error is significant either at long range or short range. If the angular uncertainty is high the error grows with the distance, as shown for the Surface Movement Radar (SMR).

Sensor	σ^{Ω}	σ^{Θ}	UST	WPL	IT.
Surface Movement Radar (SMR)	10.00 m	0.050 rad	30%	$> 30\%$	0
Universal Medium Range Radar (UMRR)	00.60 m	0.005 rad	$< 1\%$	$> 1\%$	0
Laser Scanner	00.01 m	0.001 rad	$< 0.5\%$	$> 0.5\%$	0
Surface Movement Radar Improved (SMRi)	02.00 m	0.050 deg	$< 0.3\%$	$> 0.3\%$	0

Table 3.1: Comparison of the average error induced by different transforms for different sensors. IT denotes the inverse transform error, WPL the working point linearization error, and UST the unscented transform error.

3.1.3 Temporal Alignment

Temporal alignment of data from sensors and other sources is a topic not to be underestimated. Safety features in full scale airport surveillance systems for major where useless or even dangerous, because of the incorrect temporal alignment of data from multiple sensors [110]. The means chosen to align data temporarily depend on the application scenario and the architecture of the system embedding the data fusion functionality. In some cases it is possible to infer the temporal alignment from sensor data without any information on the time of the measurement, e.g. when tracking exactly one continuously moving object with two sensors measuring the position of the object, the data of both sensors can be aligned by probabilistic plausibility checks. But these cases are very specific and do not apply to all tracking scenarios.
Therefore sensor data and other relevant data used for tracking contains a time-stamp. In the optimal case the time-stamps are accurate, flawless, and synchronized allowing the

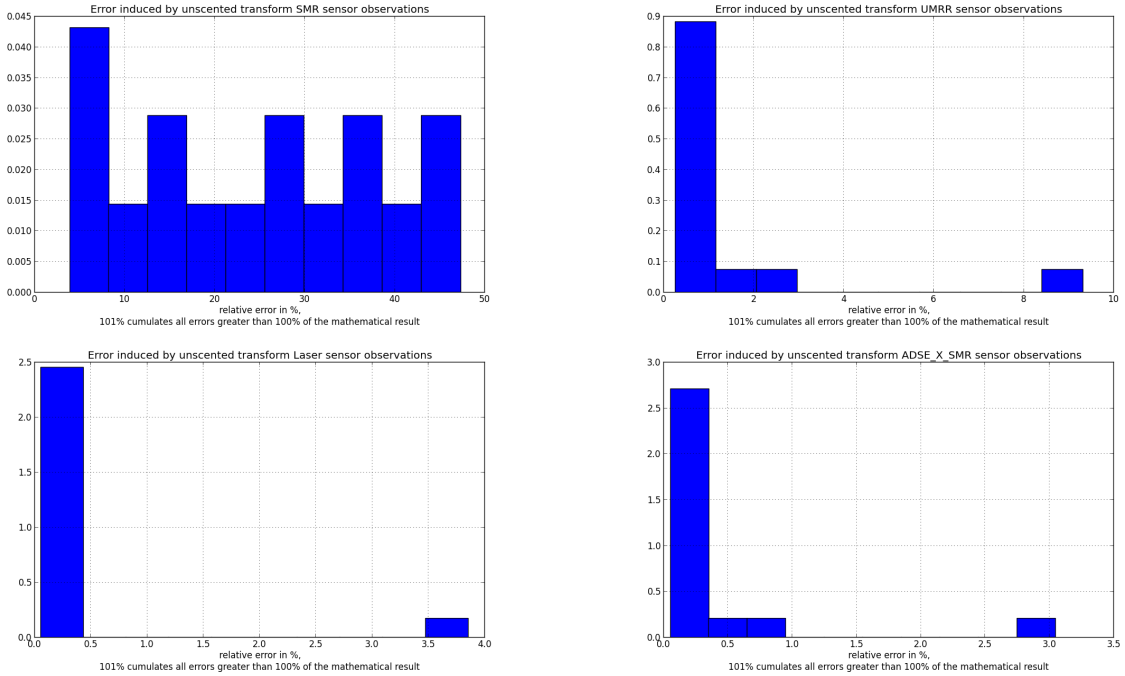


Figure 3.10: The images shows the distribution of the error with respect to the magnitude of the error. The plots are normed to an area of 1.0.

exact temporal alignment of the data. For scenarios where this is not possible algorithms as the Lamport time can be applied to partially reconstruct the temporal alignment of the data. So called Babylonian scenarios, where a data source fakes time-stamps, are the worst case scenarios. Although this case appears to be very unlikely it is a problem frequently encountered in airport traffic surveillance, and it has led to severe problems with airport traffic surveillance systems [53]. These systematically flawed time-stamps can be rooted to circumstances and technological incompatibilities frequently encountered at airports, see [109] for details.
For reasons of time and because of the broad range of application scenarios and the circumstances of the primary research objectives ensuring the temporal alignment is no topic that needs to be discussed in depth in this work. The developed design supports the temporal alignment, and the prototypical implementation provides the necessary support for the temporal alignment in the application scenarios relevant to the research objectives.

3.1.4 Data Fusion

The previous part of this section dealt with the prerequisites for a successful data fusion. Now the fusion of the data from multiple times and sources will be discussed. As stated before in Section 2.1 the goal of the fusion is "to combine elements of raw data from different sources into a single set of meaningful information that is of greater benefit than the sum of its contributing parts" and achieve the following synergic effects:

1. Representation: The information obtained during the fusion process or at the end of the fusion process has an abstract level or a granularity higher than each input

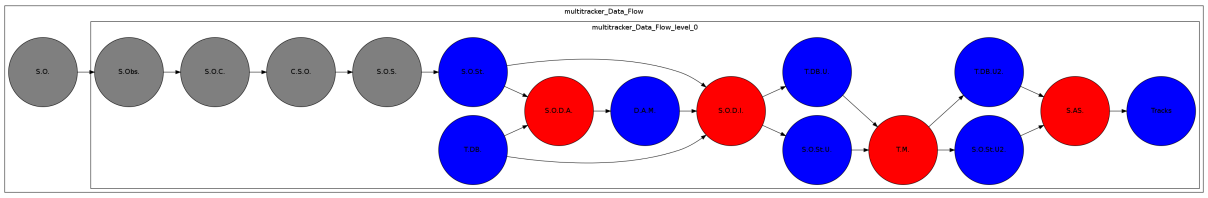


Figure 3.11: Localization of the core data fusion process, the data integration, in the refined problem graph. The parts of the graph that are primary concerned with the data fusion are colored.

data set. The new abstract level or the new granularity provides richer semantic on the data than each initial source of information.

2. Certainty: If V is the sensor data before fusion and $p(V)$ is the a prior probability of the data before fusion, then the gain in certainty is the growth in $p(V)$ after fusion. If V_F denotes data after fusion, then we expect $p(V_F) > p(V)$. [3]

3. Accuracy: The standard derivation on the data after the fusion process is smaller than the standard derivation provided directly by the sources. If data is noisy or erroneous, the fusion process tries to eliminate noise and errors. In general, the gain in accuracy and the gain in certainty are correlated.

4. Completeness: Bringing new information to the current knowledge on an environment allows a more complete view on this environment. In general, if the information is redundant and concordant, there could also be a gain in accuracy.

The benefits that apply, with respect to these synergic effects of data fusion, to tracking will be discussed in the following.

Representation: While the information from different sources like sensors, background knowledge, or auxiliary information like traffic databases only capture a part of the state of the target or the environment, and the data itself is most times not associated to any individual object, the fusion process results in a meaningful information in the context of the application. For example a radar provides a measure of reflection for each angle and distance, an information that is without further processing hardly useable for tracking objects in a cluttered environment like airport ground traffic. At the airport swaying grass, birds, buildings, rain, and multi path reflections affect the readings of the sensor in such a way that the data hardly interpretable by a human, even when it is displayed as an image showing the reflection intensity. When fused with data from other sensors like more radar sensors, background knowledge like maps, and auxiliary information like the previous sensor data, it is possible to derive the meaningful information which vehicles are at the airport, their positions, their past, current, and future movement, and their schedule from the fused data. Another example is the multilateration radar, where each sensor measures the run time of a signal from an active transponder at the vehicle to the

[3] This seems to be not true taken by word, obviously the probability of the sensor data before the fusion is the same as after the fusion, but after the fusion the belief over the state of the world will much more resemble the true state of the world.

receiver of the sensor. While the single run time is often useless, multiple receivers (often up to 30 at major airports) allow the localization of the transponders position with some accuracy,[77] and [78].

Certainty: Almost all sensors used for tracking suffer from significant random noise errors and systematic errors. These errors may be caused by the environment, inferences of the sensor with other parts of the system the sensor is embedded in, or the sensor itself. Auxiliary information like traffic databases are also prone to errors, and background knowledge is often incomplete or inaccurate. This is a situation where the data from a single source is not only incomplete but also uncertain. For example one cannot be sure whether the reflection measured by a radar is caused by environmental conditions that were not meant to be measured or by a vehicle reflecting the radar beam; Pixels in a camera image may be affected by electronic inference during the conversion of the analog charges of the photosensitive cells to digital numbers. The fusion of the data like for example filtering a neighborhood of pixels, or combination of data from multiple radar sensors makes it possible to estimate the likelihood of the single data much better, resulting in a higher certainty. This is also true when fusing data over time, here radar reflection centers that exhibit a , with respect to the application scenario, sensible behavior, can be assumed to be a vehicle with much more certainty than is possible from a single measurement.

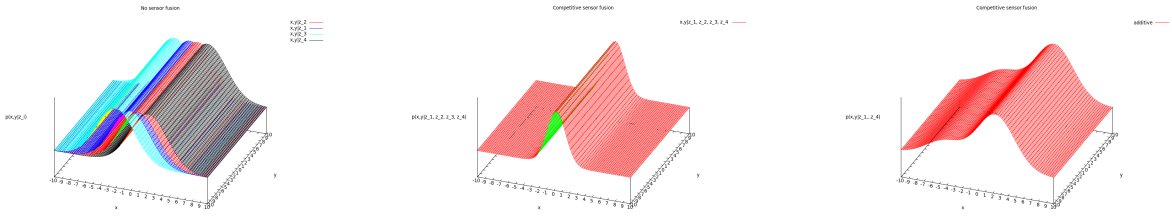


Figure 3.12: Accuracy: The fusion of the competitive measurements, shown in the left plot, with Bayesian method, shown in the middle plot, and additive method, shown in the right plot, leads to a more accurate estimate of the variable then the individual measurements.

Accuracy: As sensors suffer from random and systematic errors, each measurement or observation is more or less prone to these errors and therefore more or less inaccurate. Multiple measurements or observations, be it from multiple sources or multiple times can, under favorable circumstances, be combined to a much more accurate information. Figure 3.12 shows how the simple multiplication or addition of sensor observation probability densities can increase the accuracy. Accuracy enhancing data fusion is achieved using competitive sensor setups that measure the same quality of the environment.

Completeness: Most sensors or other sources of data provide only a fraction of the desired information on a tracked object's state. Completeness is often achieved with so called complementary sensor setup that measure different qualities of the environment.

Figure 3.13 shows the fusion of two independent measurements, one measuring the x-translation of an object and the other its y-translation using Bayes law.

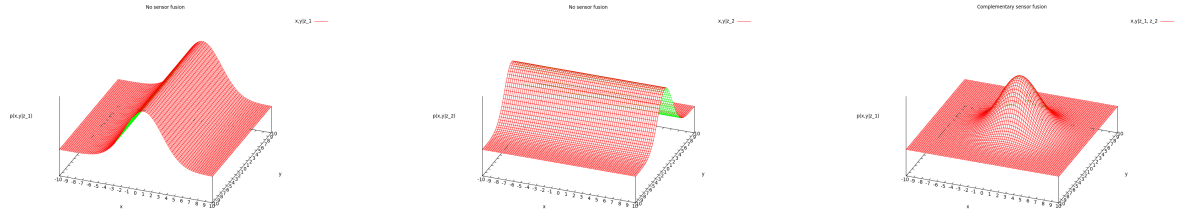


Figure 3.13: Completeness: The combination of the complementary measurements of the x variable, shown in the left plot, and the y variable, shown in the middle plot, results in more complete estimate, shown in the right plot, of the state of the joint variables then any of the single measurements.

3.1.5 Data Fusion Methods and Algorithms

The following text in the section discusses the means to reach this goal that is the mathematical models, the algorithms and the formal architecture of the system. There exists a broad range of approaches that show reasonable performance. Figure 3.14 shows the difference of two different data fusion methods. Although booth methods have a good performance, their data fusion results show a significant difference. While the Bayesian fusion is mathematically sound it does not necessarily outperform other methods if the model doesn't capture the environmental mechanics good enough.

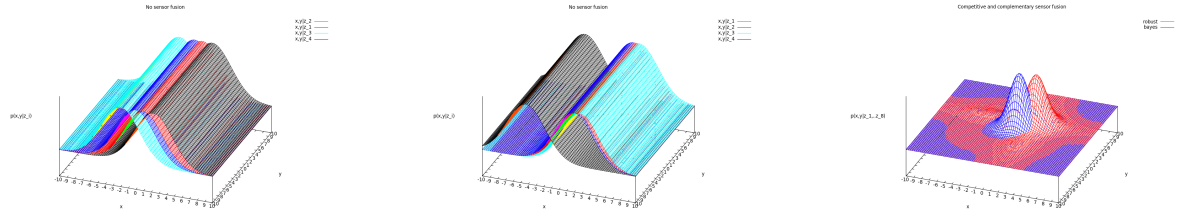


Figure 3.14: Comparison: The fusion of the competitive and complementary measurements of the x and y variable, shown in the left plot and the middle plot, results in a more accurate and complete estimate of the state of the variables. As visible in the right plot, depending on the fusion method, the result can be quite different.

Tracking is a process, which integrates sensor data that represents measurements of specific qualities of the environment at discrete[4] times, over time. Therefore it is necessary to model the change of the environment between the measurement points. Besides Bayesian laws that provide a mathematically sound foundation, there exist a wealth of data fusion methods, ranging from logical inference to neural networks that have been shown to be

[4]Of course the measurements measure over a certain duration, but these are usually so small that it is legal to refer to this periods as points.

able to handle tracking task. Unfortunately many of these are not suited for the realization as an embedded system. Reasons for this are manifold and will not be discussed further at this point.
The important fact is that there is a certain degree of orthogonality in the concepts and task involved in the tracking process. A well designed architecture can reflect this orthogonality in the independence of the (formal) process architecture. To derive such an architecture it is useful to start with the most basic model of a sensible digital representation of a dynamic environment. This model is based on the following assumptions.

- The current state of the world x_t at time t is directly caused only by the past state of the world x_{t-1} at time $t-1$, so the future state of the world x_{t+1} at time $t+1$ will be directly caused only by the state of the world at time t.
- Usually one cannot assess the state of the world directly, but rather perceive perceptions z_t

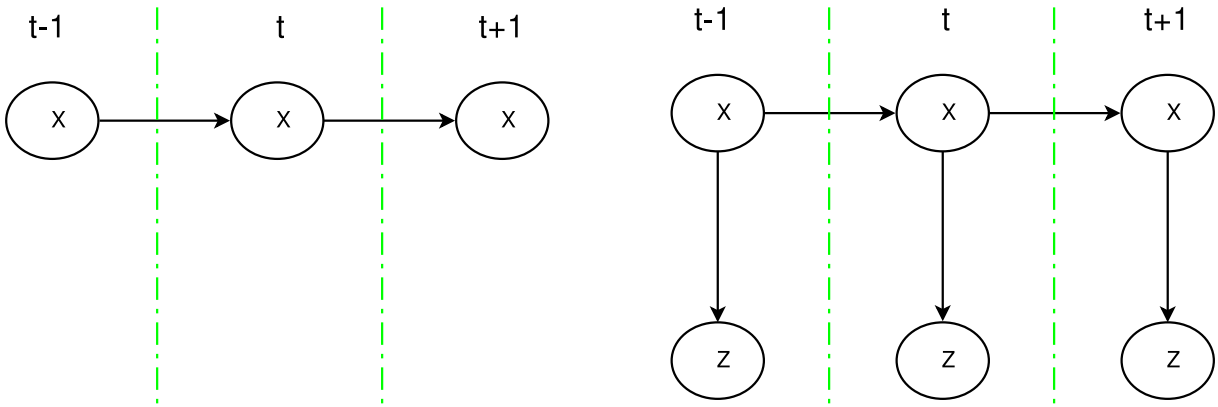


Figure 3.15: Model for discrete data fusion. The left image show the basic net based on the sensible assumption that the state of the world x at time t only depends on the state of the world before, at time $t-1$. The right image adds an observation model where each observation z only depend on the state of the world.

This model is shown in the diagrams of Figure 3.15, which is the basic model is the basis for a vast range of filtering algorithms, as well as for full scaled architecture concepts. In order to provide sufficient results this approach requires that the model that represents the world captures the state of the world in such a way that everything that directly affects the dynamic state of the world significantly is included in the representation of the state of the world. Furthermore the transition function that determines the relation between the current state of the world and the past state of the world must use that information so that the information contained in the current state of the world is self sufficient.
There exists a, for many embedded system sensible, extension that includes the actions of the system itself as these are usually known and not perceived, as well as their direct consequences.

- The state of the world is affected by ones own actions u and we can predict the direct consequences with sufficient accuracy.

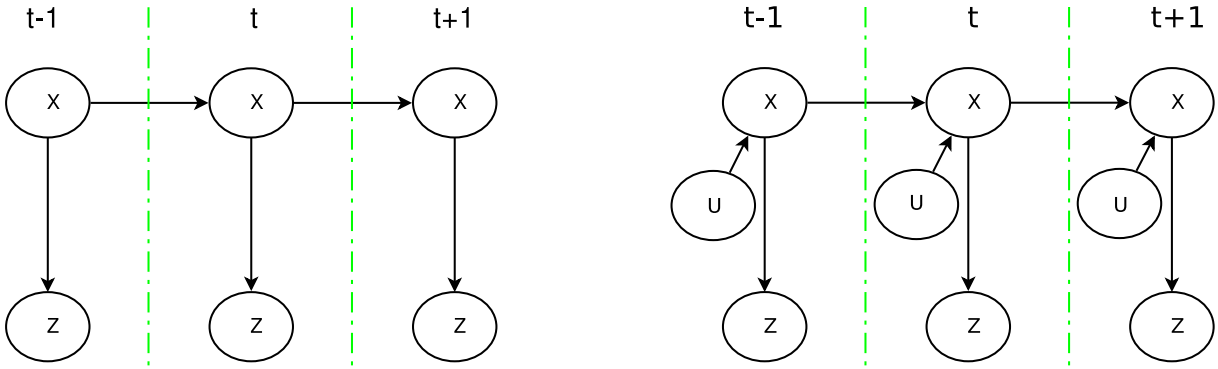


Figure 3.16: Extended discrete word model. In the right image information about actions u that are known rather than perceived have been added, to the model shown in the left image. The current world state is modeled as a consequence of the past world state and the current actions.

- In this case the current state of the world x_t, depends directly only on the state of the world before and one's actions at that time u_t.

The resulting model is shown in Figure 3.16. Like its predecessor, the model is well suited as a basis for data fusion algorithms and architectures. Orthogonal concepts are now accessible, the only formal requirement of the transition function that computes the next world state is that it accepts the previous world state x_{t-1} and the control information u_t as input and provides the current world state x_t as output. Therefore this function is completely independent of the function that models the relation between the state of the world and the perception z of the world, and its inverse that models what the state of the world is given the perception of the world. With increasingly finer granularity of the model more orthogonality emerge that, if resembled by the architecture, allow a large degree of freedom in the design of the individual architectural components. As a result software or hardware implementations of these architecture can be very flexible, because they have a modular design that allows the configuration of the system from a set of solutions for the individual tasks.

Bayes Filter

To derive the Bayes filter convert the model shown in Figure 3.15 or that shown in Figure 3.16 into a dynamic Bayesian network. The Bayes filter algorithm, as shown in listing Algorithm Bayes Filter (X_{t-1}, Z_t), can be run from any time t_0. The algorithm performs an iterative processing of the recursive mathematical definition of the filter as the dynamic network grows as time proceeds.

Algorithm Bayes Filter (X_{t-1}, Z_t):

1 $\overline{X_t} = \int p(x_t|X_{t-1})dx$;
2 $X_t = \int p(\overline{x}_t|Z_t)d\overline{x}$;
3 return X_t;

The recursive mathematical definition is that the current state of the world X_t is defined

by the past states of the world $X_{t-1}, \ldots, X_0$ so the probability distribution $P(X_t)$ is defined by the equation

$$P(X_t) = P(X_t|X_{t-1}, \ldots, X_0) \tag{3.7}$$

With respect to the first order Markov assumption explained in the preceding text - and shown in the left image in Figure 3.15 - the equation resolves to

$$P(X_t) = P(X_t|X_{t-1})P(X_{t-1}) = \alpha \int p(x_t|X_{t-1})P(X_{t-1})dx_t \tag{3.8}$$

with the normalizing constant α that ensures that $P(X_t)$ integrates to 1 and

$$p(x_t|X_{t-1})P(X_{t-1}) = \alpha \int p(x_t|x_{t-1})p(x_{t-1})dx_{t-1} \tag{3.9}$$

If one does not know the state of the world but has to rely on measurements, the actual problem is to determine the current state of the world given the observations of the current state and past states of the world, with the equation

$$P(X_t) = P(X_t|Z_t, \ldots, Z_0) \tag{3.10}$$

The assumption of the conditional independence of the observations $Z_t, \ldots, Z_0$, which was explained in the preceding text and shown in the right image in Figure 3.15, and the application of Bayes law to calculate $P(X_t|Z_t)$ results in the Bayes filter algorithm. The initial distribution $P(X_0)$ is usually assumed to be even, or it is based on background knowledge and/or auxiliary information.
Bayes filters can be generally classified into parametric and non parametric filters. Parametric filters use parametric representations to represent probability distributions. Parametric representations use a parametrized mathematical function or model with a fixed number of parameters for the representation of objects or distributions and assume a fixed structure of the model. Non parametric filters use non parametric representations to represent probability distributions. Non parametric representations have a flexible number of parameters that depends on what one wants to represent and the structure is determined from the given data. While parametric filters are often computationally efficient for linear filtering tasks, non parametric filters can handle non linear system behavior much better, often at the cost of additional computational resources. The larger amount of computational resources needed by non parametric filters result from the fact that the representation of the state of the world X must be represented by a sufficiently large set of variables x that represent the different possible states of the world with sufficient precision and coverage. For typical tracking and localization tasks the following popular Bayesian filters are available:

- parametric:
  - Kalman Filter
  - Information Filter
- non parametric:
  - Histogram Filter

– Particle Filter

From these the particle filter is of particular interest for the application scenarios, because it can approximate arbitrary probability density functions. The particle filter allows the implementation of very complex state transition functions that can not be realized using other approaches. Compared to a histogram filter the particle filter is more efficient in terms of accuracy when using the same computational resources, because it concentrates the computational resources at areas of high probability allowing a more accurate estimation of the systems state.

Particle Filter

The particle filter is a nonparametric Bayesian filter. This class of Bayes filters does not rely on a fixed functional form, like normal distributions, of the posterior. The probability distribution is represented by a finite number of values, in the case of the particle filter samples. Each sample refers to a specific point in the state space of the systems state. For tracking ground traffic tasks usually the position, heading and the first derivative. The algorithm of the particle filter is explained in detail in Section 4.

3.1.6 Tracking Multiple Objects

Tracking only one object does not meet the requirements of many robotic tracking applications that require dealing with persons, objects and other robots. Even a lot of tasks that do not explicitly require the tracking of multiple objects can be done more efficiently by a robot that is able to track multiple objects. Tracking multiple objects is a demanding task, not only from a mathematical point of view but also with respect to the computational resources and time needed for the execution of the tracking algorithms, and includes problems where a proper mathematical treatment can be shown to be NP-Hard, , see Appendix C. To efficiently track multiple objects the Bayes filter can be extended on the basis of some weak assumptions on the conditional independence of the objects behavior.

Adoption of the Bayes Filter Model to Multi Object Tracking

When tracking multiple objects with the Bayes filter the formal model can be treated as a model where the state of the world X_t is a vector $X_t = \{x_t^1, \ldots, x_t^n\}$ of the states of the objects in the world one is interested in, see Figure 3.17. The perception of the world Z_t is a vector $Z_t = \{z_t^1, \ldots, z_t^m\}$ of observations from different sources. If the state, in this case the movement in the immediate future, of an object does not significantly affect the state of other objects it is possible to divide the task of tracking multiple objects.

Challenges specific to Multi Object Tracking

Tracking multiple objects is generally far more complicated than tracking a single object. Circumstances usually encountered in real world scenarios that contribute to this are:

- Multiple objects of interest: There is more than one object of interest in the environment, and the exact number of objects in the environment is unknown and changes over time.

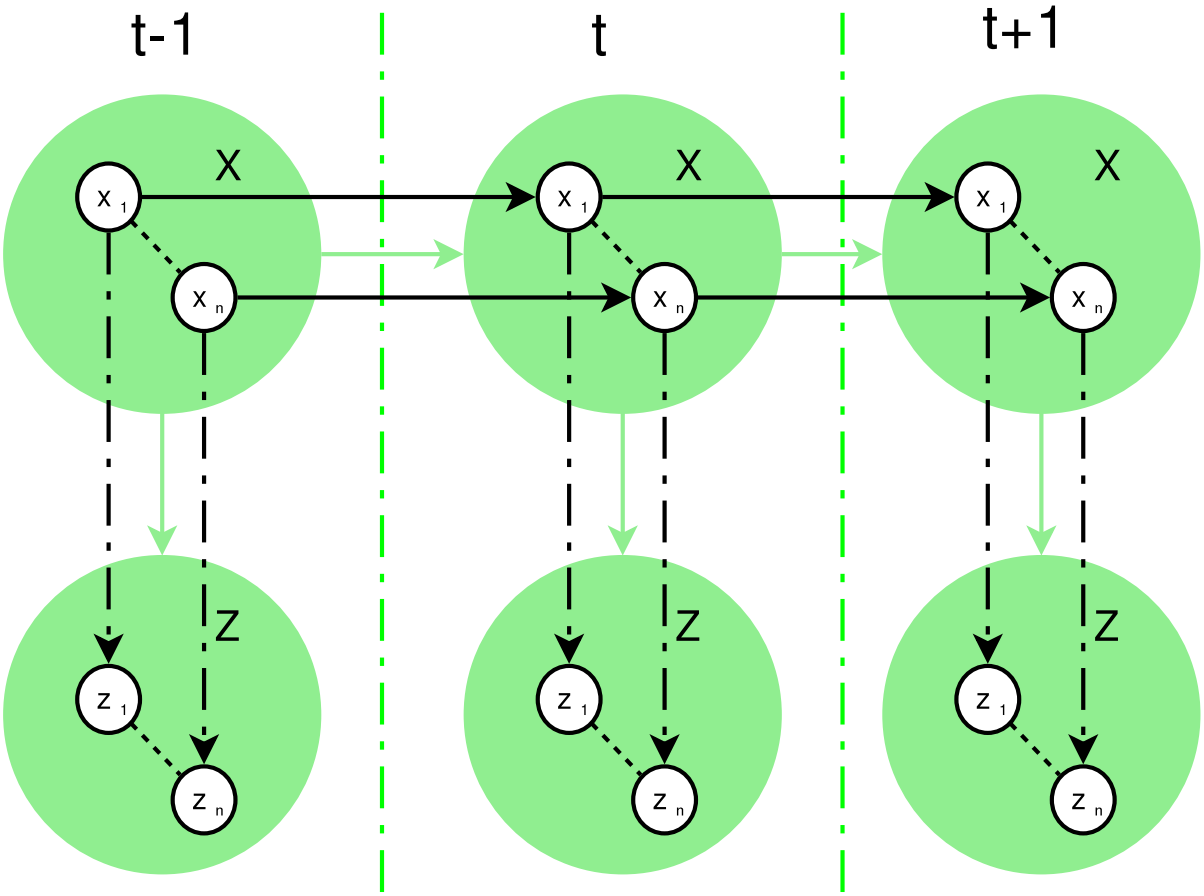


Figure 3.17: The Bayes filter applied to multiple object tracking. The assumption of the conditional independence of the tracked objects allows the division of the traffic situation estimation problem into many object state estimation problem.

- Different and unknown types of object: Often the tracked objects are of different types and behave different. Depending on the difference it can be impossible to accurately model the relationship between past, current, and future state of the objects with a general function.

Often the environment is also cluttered. In the context of tracking multiple targets a cluttered environment refer to an environment that is characterized by the following qualities, which complicate the tracking task.

- (Partial) Occlusion: Objects of interest may be occluded by other objects of interest or parts of the environment. The occlusion may be fully or partially for an arbitrary duration.
- False Alarms: The environment or errors in the sensor technology cause systematic or random sensor measurements that are wrongly classified as a objects of interest.
- Object Inference: Objects may infer with each other, which affects the dynamic behavior of the objects in such a way that the complexity of the computation of an appropriate mathematical model is intractable.

- Unidentified Observations: The measurements of different objects of interest do not have a characteristic signature that allows the ambiguous mapping of sensor measurements to objects.
- Uncertainty of Sensor Measurements: The complexity of the environment leads to systematic and random errors that result in a deviation between the measured state of the world and the true state of the world.

In order to deal with these circumstances the tasks Data Association, Track Management, and Observation Management, which will be explained in detail later in the text, are usually part of the task of tracking multiple object.

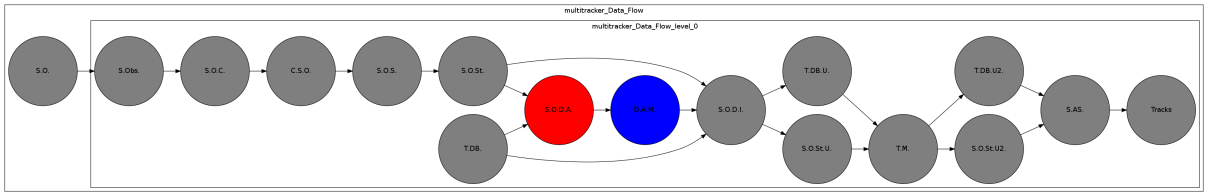


Figure 3.18: Localization of the data association in the problem graph.

Data Association: One of the most frequent[5] problems encountered when tracking multiple objects is the correspondence between objects and observations. When the task of tracking multiple objects is divided into multiple task of tracking single objects, it is important to know which observations are caused by which object. Many sensors do not provide this information, so a part of the dynamic Bayesian network that describes the task is hidden, see Figure 3.19. With respect to the different levels of abstraction, association is optimally done on the level of signals, situations, and most frequently features. Like many tasks related to data fusion data association is not a trivial task, which is among other difficulties caused by the computational complexity. The exact solution of the problem can be shown to be NP-hard, see Appendix C.
Although the problem is not new and is topic of established books, for example the authors of [94] state that "With n observations and n tracks... ...there a $n!$ possible assignments of observations to tracks; a proper probabilistic treatment must take all of them into account, and this can be shown to be NP-hard". By 2001 the problem was considered to be solved generally, but Uhlman [41] stated that "Over the years, many attempts have been made to devise an algorithm for multiple-target tracking with better than $O(n^2)$ performance. Some of the proposals offered significant improvements in special circumstances or for certain instances of the multiple-target tracking problem, but they

[5]The estimation of the combined and individual state of multiple objects does not necessary require data association, as the following statement of Hall et al shows "Association is not an essential ingredient in combining multiple pieces of data. Recent work in random set models of data fusion provides generalizations that allow state estimation of multiple targets without explicit report to target association." [41] . Also a broad range of methods from the field of artificial intelligence is available that can - based on noisy observations - provide robust estimates of a systems state without the need to explicitly associate the data. Machine learning algorithms such as recurrent neural networks can reconstruct the state of a system from noisy and partial data.

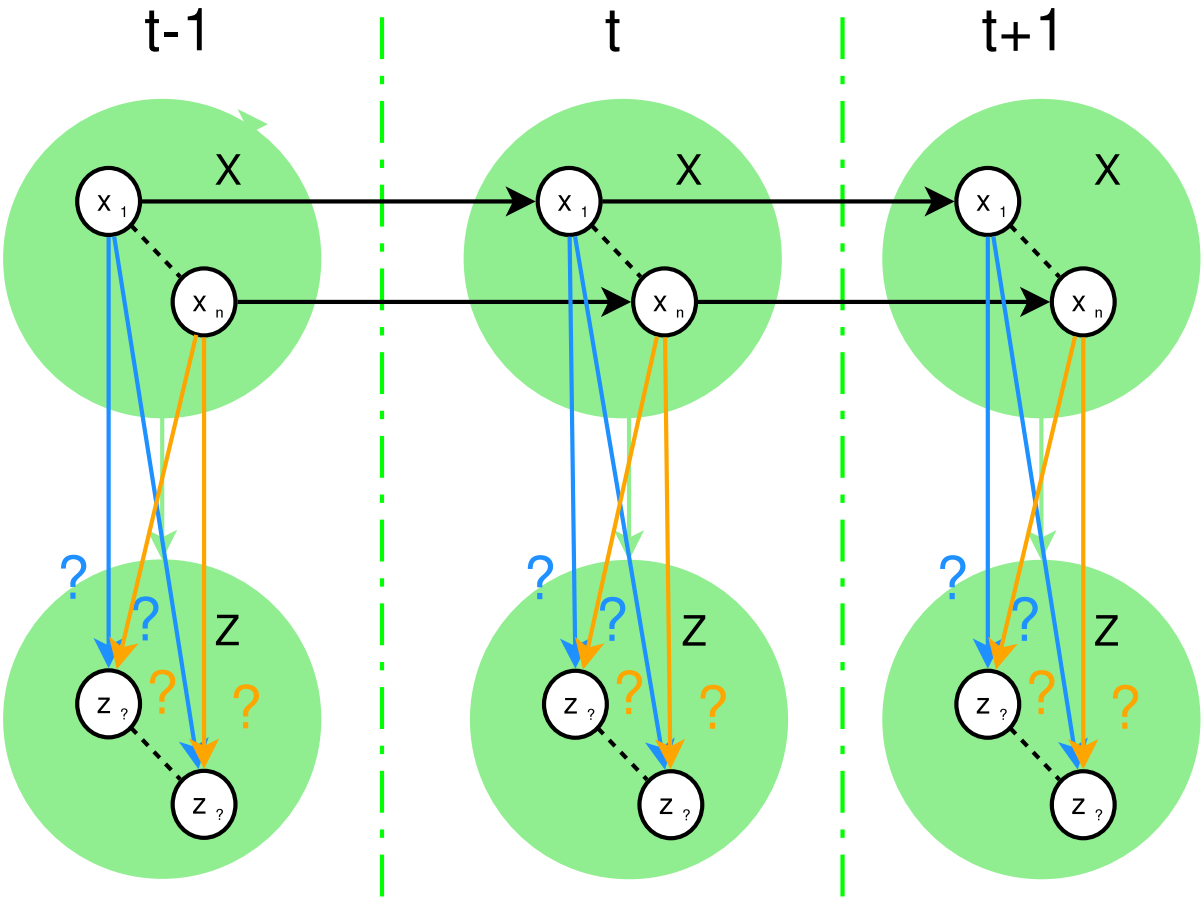


Figure 3.19: The data association problem. In many application scenarios sensors cannot identify objects. Therefore it is not known which observation belongs to which object.

retained their $O(n^2)$ worst-case behavior..... Even with the new methods, multiple-target tracking remains a complex task that strains the capacity of the largest and fastest supercomputers." Data association is important for this work because the approach to track objects individually, which supports a distributed and hierarchical architecture, requires an explicit data association. Thus the system must support data association related functionality at multiple locations, a constraint that strongly influences the design of the tracking system. Also the concurrent computation of the data association reduces the time complexity a lot as shown in the Section 3.2.2.

Track Management: An important task is the management of tracks. Based on the past and current information the number of tracks must be estimated, tracks that have left the area must be deleted, and new tracks that appear must be instantiated.
There exist numerous approaches and track management is often closely coupled to observation management, which is discussed right after track management. Most popular approaches are probabilistic and use the overall number of observations or the number of observations per sensor in a fixed time interval to determine the most likely number of tracks. Another popular approach is to collect observations that could not be associated with any existing track and infer tracks from these observations; often with additional

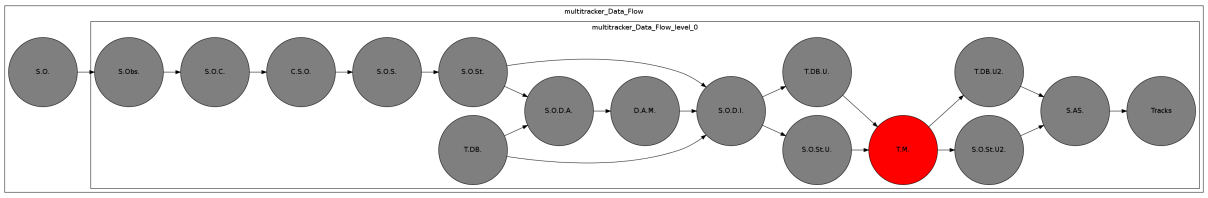


Figure 3.20: The track management in the problem graph.

plausibility checks, and regression solving to instantiate new tracks with parameters inferred from the observations, e.g. heading and speed from a set of position measurements.

Observation Management: The management of the incoming sensor observations is very important. Incoming sensor observations should be stored in a database for a number of reasons.

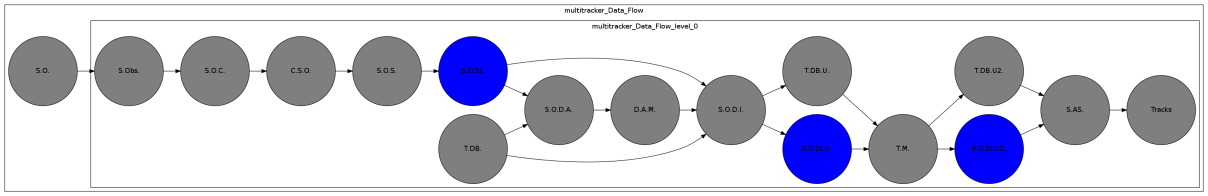


Figure 3.21: Observation management in the problem graph.

- Delay: Different delays during transmission cause sensor observations to be received unordered, and the newest observations may not always be the most accurate. In some cases it is therefore recommendable to update the track by integrating late sensor observations.
- Data Association: In a broad range of tracking tasks it it sensible not to associate observations as soon as available, but rather consider a set of observations and tracks to find the most likely association for the observations.
- Estimating the number of tracks: The number of observations made by sensors is usually closely related to the number of objects in the area. Depending on the circumstances in the application scenario, the number of objects may be directly inferred from the number of observations, or the likelihood of an untracked object can be estimated from observations that could not be assigned to a track.
- Instantiating new tracks: When a new track is instantiated, the object has usually caused some sensor observations that where not assigned already. The new track can be instantiated more accurately based on these observations. A well establish paradigm is "track before report". Unassociated sensor observations are searched for meaningful patterns that could be caused by an object. If the likelihood for a new track exceeds a threshold the new track is instantiated.
- Improvement of quality: The quality of tracking results can be improved using a technique know as smoothing, where the state of a tracked object is inferred by past

and future measurements of the object. It is also a common approach to use stored observations to counter effects, which are caused by deficiencies of the mathematical models of the observed processes in particle filters, by rerunning the algorithm on a part of the sample set with past measurements. The deficiencies are usually related to the 1. order Markov chain assumption that is not fulfilled because the mathematical model does not capture all aspects of the tracked objects state with sufficient accuracy.

3.1.7 Situation Assessment

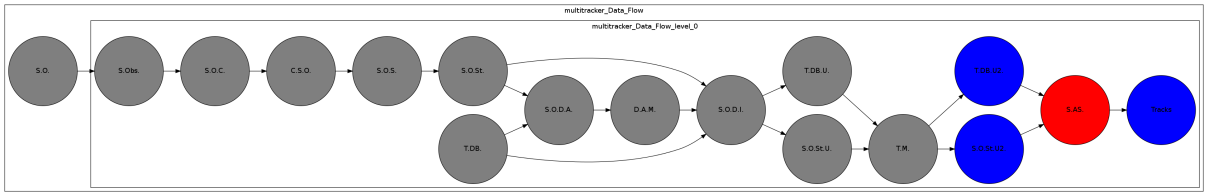


Figure 3.22: Localization of the situation assessment in the problem graph

To be of use for humans or superordinate technological instances the meaningful information, which results from the data integration process, should be presented in a compact form. Also information that is not deductible from single track data should be deduced during this stage of the data fusion process when considering all tracks, background knowledge and auxiliary information. Such information includes, but is not limited to: Imminent conflicts between tracked vehicles, dangerous situations, or the violations of no go areas. The result of the situation assessment is a list of extended tracks that contains a compact representation of the probability density function over the tracked vehicles state, and also contains auxiliary information on the tracks that is valuable in the specific application context of the tracking system.

3.2 Hardware-Software Partitioning

This section discusses the theory of the design, development, and implementation of an embedded tracking system, consisting of hardware and software components, for application in scenarios from the robotics, airport, or automotive field. The conceptual design of the architecture is introduced and motivated based on the requirements and specification of the precedent mentioned application scenarios and the theory introduced in the previous section.
Formally, the problem is to map the problem graph to an architecture graph of the systems hardware in such a way that the resulting system is well suited for the majority of application scenarios. The basic mapping as shown in Figure 2.10, must be refined to determine which parts of the data fusion process are realized in software and which parts are realized in hardware. The resulting mapping, as shown in Figure 2.12, is motivated and explained in the rest of this section.
The refined problem graph is shown in Figure 3.2, and the refined architecture graph is shown in Figure 3.23; the system architecture is made from 3 functional nodes: The

Sensor node, the pc104 node, and the FPGA_Tracker node. These are connected by 2 communication nodes: The bus node that connects the sensor node and the pc104 node and the pc1042fpga_bus that connects the pc104 node with the FPGA_Tracker node.

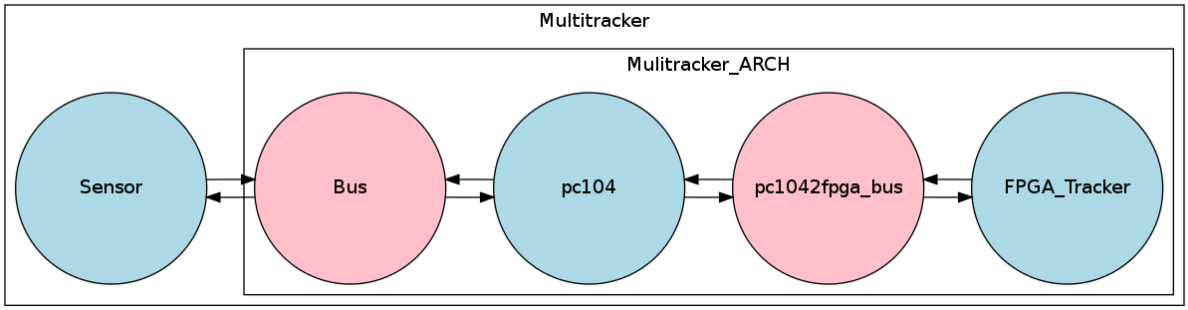


Figure 3.23: Refined architecture graph of the multi tracker system.

The system architecture is a good basis for a research and development prototype for of the following reasons. First, the pc104 node represents an industrial standard pc that allows the convenient integration of additional peripheral components and provides plenty of interfaces to attach sensors to the system, while being robust enough to be deployed in a real application scenario. Also the pc104 can be replaced by a desktop pc outside the application environment to allow for better research and development working conditions. With the pc is possible to test the functionality and integration of hardware in the system, by software components that mimic the hardware. Second a FPGA, represented by the FPGA_Tracker node, is a very flexible yet powerful choice to implement algorithms in hardware with a semi custom design style. FPGA development boards that support a reasonable range of additional peripheral hardware are available from most major FPGA distributors along with integrated development environments (IDEs). System builders that provide soft core processors, which allow the convenient design of systems on a chip including custom logic components, are readily available for these IDEs. FPGAs can be configured trough the combination of different design approaches, and the developed designs can be used as a basis for a full custom design.

3.2.1 Requirements & Specifications

Based on the target application scenarios the following requirements have been defined for the tracking system.

- System Integration: The tracking system should be easy to integrate into existing mobile or fixed platforms like robots, cars, airport infrastructure, and sensors.
- Tracking Accuracy: The system performance in terms of accuracy must be on par with state of the art software solutions.
- Tracking Speed: The system must be able to integrate sensor data and provide results in compliance to hard real time constraints. For most of the application scenarios these constraints require a sensor data integration rate of at least 30Hz.

- Power Consumption: The power consumption of the system should be very low compared to software solutions.
- Heat Dissipation: The system must operate under extreme environmental condition and will often be placed in a housing with very limited heat exchange capabilities. Therefore the system should produce a minimum of heat.
- Size: The system should be small enough to be integrated into small humanoid robots or embedded sensors as encountered in the automotive field.
- Reliability: The systems formal models, algorithms, and hardware should be robust in such a way that the system can provide its functionality under the environmental conditions encountered in reasonable application scenarios.
- Maintenance: The system should be easy to maintain. Hot swaps of components should be possible.
- Safety: The system should not harmfully infer with other systems. The system must be capable of self diagnostic functions and indicate failures to superordinate systems and personnel.
- Flexibility: The system should be flexible enough to adopt to new circumstances, data sources or changes in the environment - or system - it is embedded within.

The lower bound for the fulfillment of the above requirements on tracking accuracy, reliability, flexibility, and system integration is set by the formal models and algorithms used for the tracking. Shortcomings of the formal models and the algorithms cannot be compensated for by the performance of hardware or software implementation of these. Therefore the fundamental decision is about the right tracking algorithms and right mathematical models. From the wealth of models and algorithms available Bayes filter are best suited to provide the flexible, transparent, and sound mathematical background that allow the system to be adopted to the circumstances encountered in the broad range of possible application scenarios. From the Bayes filter algorithms the particle filter is the one that has been shown to be successfully applicable to a broad range of scenarios, without major modifications of the core algorithm and the underlying mathematical models.
In the context of tracking multiple objects a particle filter can be used to track a single object, calculate independent data association probabilities, and integrate data from sensors and other sources. The particle filter's intrinsic ability to approximate arbitrary probability density functions and in-cooperate non linear models, allow the adoption of the particle filter to almost any tracking scenario without a significant changes of the algorithm. Furthermore the particle filter can be directly used to calculate the independent data association probability, and it allows the integration of sensor data with a full joint data association probability. With the full joint data association probability the individual tracks are updated with respect to all possible data associations.
Although extensions of the Kalman filter have been developed that can handle a limited degree of non-linearity and allow the appliance of the joint probability data association [4], these algorithms usually require much more tuning, to be fitted to the application scenarios needs, than the particle filter. The major drawback of the particle filter, when

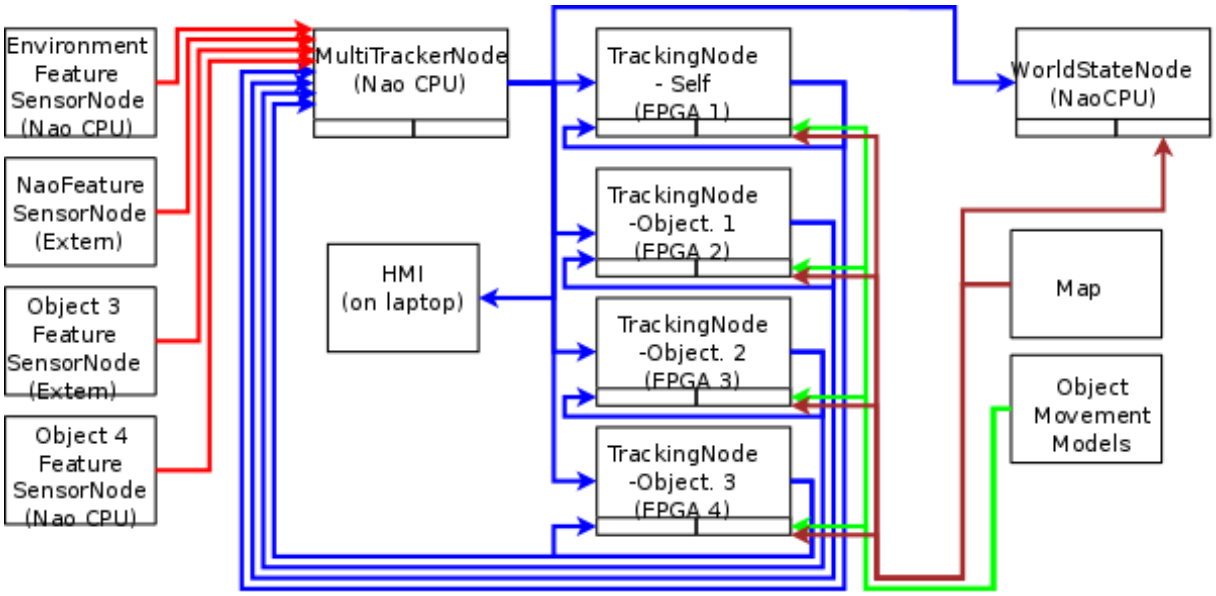


Figure 3.24: Example formal architecture of a tracking system for a small autonomous soccer playing robot.

compared to other Bayesian filter variants, is the computational complexity of the operations needed during the calculation of the data association and data integration.
In order to find a suitable partitioning into hardware and software the time complexity of the execution of the most time consuming part of tracking multiple object is compared for a software only and a hardware only solution. The comparison is done under the assumption that only custom hardware can provide the concurrent computation of multiple particle filters under the constraints that apply from the requirements on power consumption, heat dissipation, and size. The formal architecture used for the estimations might for example look like the one shown in Figure 3.24 that was developed for small humanoid soccer playing robots. It supports methods for observation management, track management, data association, and data integration as introduced in Section 3.1.

3.2.2 Performance Estimation of Software vs Hardware and Software

The computationally most expensive part of tracking multiple objects with multiple particle filters is the data association and data integration. Figure 3.25 shows a flow chart of this part of the process. While some parts of the algorithm do not benefit from a concurrent execution, the time complexity of this part of the data fusion process can be significantly decreased by the concurrent execution of the core particle filter algorithm by dedicated hardware. Here the gain in performance is not only from the execution of the core algorithm in hardware but from the concurrent execution. To account for the requirements on power consumption and heat dissipation, the clock of the hardware is set significantly lower[6] for this estimation. Also a fast data association method that assigns

[6]A comparative study found that for matrix multiplication operations the speed of a 150 MHz FPGA is comparable to that of a 1.5 GHz Pentium. Here a 50 MHz FPGA is compared to a 2.5 GHz desktop

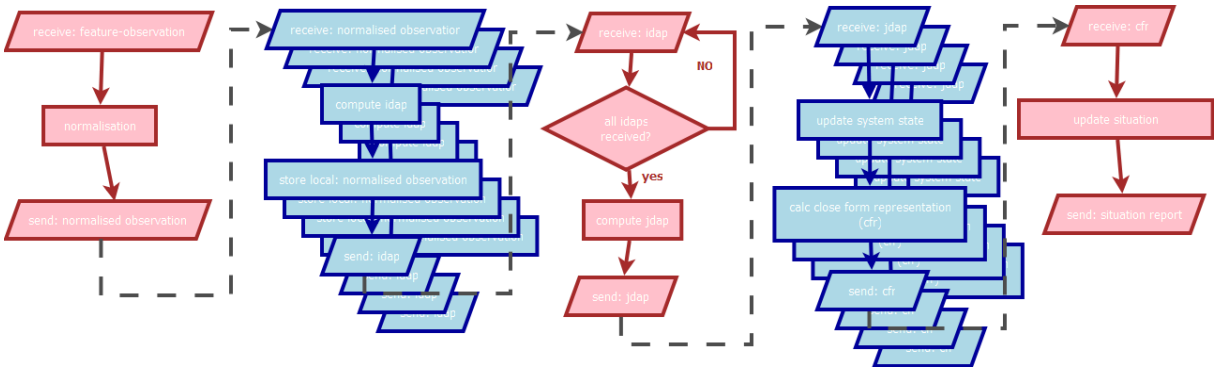


Figure 3.25: Multitracker data association and data integration flow chart. The magenta colored parts can be executed computationally efficient in a serial manner, while the concurrent execution of the blue parts allows the reduction of the time complexity of the algorithm.

maximal 1 measurement to each track is used for the estimation.
The number of samples used by each particle filter is an important factor for the time complexity of the execution. The number of samples apparently sets the limit to the accuracy and error of particle filters, which is an active field of research [24]. The performance of particle filters can be estimated in comparison to the optimal filter. The optimal filter is, except for a very restricted set of dynamic models, not expressible in a closed form. It has been shown that the upper bound on the variance of estimation error depends on the number of samples M and has the form $c \cdot O(M^{-1})$. The upper bound is independent to the number of dimensions of the sample state space, but that the constant c heavily depends on the state vector dimension [21, 19]. The grow of c is linear to the number of dimensions with a good importance density respectively proposal distribution; with a bad proposal distribution the variance of error becomes exponential to the dimension of the state vector. This "curse of dimensionality" has been experienced in many practical applications [91] and also applies to robotic applications like tracking or pose estimation. While tracking stationary objects on a 2 dimensional map requires only 3 dimensional state vector, tracking moving objects in a 3 dimensional volume requires a 12 dimensional state vector. Therefore a high number of samples is necessary to keep the error small and enable the robot to achieve its goals when dealing with complex environments like the real world.
Because the number of state space dimensions for robotic applications is often bigger than 3 the number of samples should be in the order of hundreds or thousands. The time needed to execute a software particle filter is the total time needed for data association, track management, update of the tracks, and extraction of the probability density. For the sequential execution with a software system, which is at time step t tracking N_t objects when K_t features are detected, this time is approximately given by the equation

$$K_t N_t c_{K_t} M + K_t N_t a + K_t d M + N_t z M + N_t p M \tag{3.11}$$

where M is the number of samples used by each particle filter, c_{K_t} is a constant for the time needed computing the independent data association probability of a detected feature

pc processor, which results in a ratio of app. 1 to 10. [8]

and sample, a is the computation of the data association probability with respect to all tracks and detected features, d is a constant for the time needed to update a sample given a feature, z is a constant for the time needed for the importance sampling, and p is the time needed to integrate a sample in an abstract representation of the probability density. The time needed for the execution of the algorithm strongly increases with the number of detected features and tracks.
Moving most of the computations from software to hardware allows the concurrent computation for each track. The computational architecture of the particle filter implementation as multiple hardware particle filters allows the concurrent execution of major parts of the particle filter algorithm, in particular the computational expensive computation of the particle filter algorithm. Therefore M an K_t and N_t are removed as a factor from most terms, and the time needed by this system can be approximated Equation 3.12.

$$K_t c_{hw} + K_t N_t a + d_{hw} + z_{hw}(M) + p_{hw}(M) \tag{3.12}$$

Here c_{hw} is the time needed by each hardware particle filter to calculate the independent data association probability of the track given a feature. d_{hw} is the time needed by each hardware particle filter to update the track given a feature. $z_{hw}(M)$ is the time needed each hardware particle filter for the importance sampling, and $p_{hw}(M)$ is the time needed by each hardware particle filter to extract the probability density from the sample set of the track. In comparison to the time needed by a pure software solution, as shown in (3.13), the number of tracks N_t does not play a major role anymore as does the number of samples M. Furthermore c_{hw} and d_{hw} are close to independent of the number of samples M.
This is due to the fact that the independent data association probability of a sample and the weight of a sample can be calculated independent from other samples in the sample set of the track. The prototype implements a low variance resampling algorithm that is linear to the number of samples M. The time needed for the extraction of the probability density is also linear, with respect to the number of samples M, in the prototype implementation. Fig. 3.26 shows the time needed for the execution of the algorithm by the two systems. For the calculation of the values shown in the plot constants in Figure (3.12) are 10, constants in Figure (3.13) are 1 to account for the lower clock rate of the FPGA, and the number of samples N is 1000 for both equations. K_t and N_t are given on the x-axis and y-axis of the plot.
The hardware solution outperforms the software solution for the majority of cases. Under the assumption of a full joint data association probability (FJDAP) , often desirable in robotic scenarios, the formulas need to be changed to:

$$K_t N_t c_{K_t} M + K_t N_t a + K_t N_t d M + N_t z M + N_t p M \tag{3.13}$$

for the software solution, and to

$$K_t c_{hw} + K_t N_t a + K_t d_{hw} + z_{hw}(M) + p_{hw}(M) \tag{3.14}$$

for the hardware solution. To make the estimate more robust and account for the reduced computation overhead, optimized control, and data flow of the hardware, the speed advantage of the software solution has been changed to factor 2 and the formal time complexity has been estimated also with these values. The results are shown in the lower row of

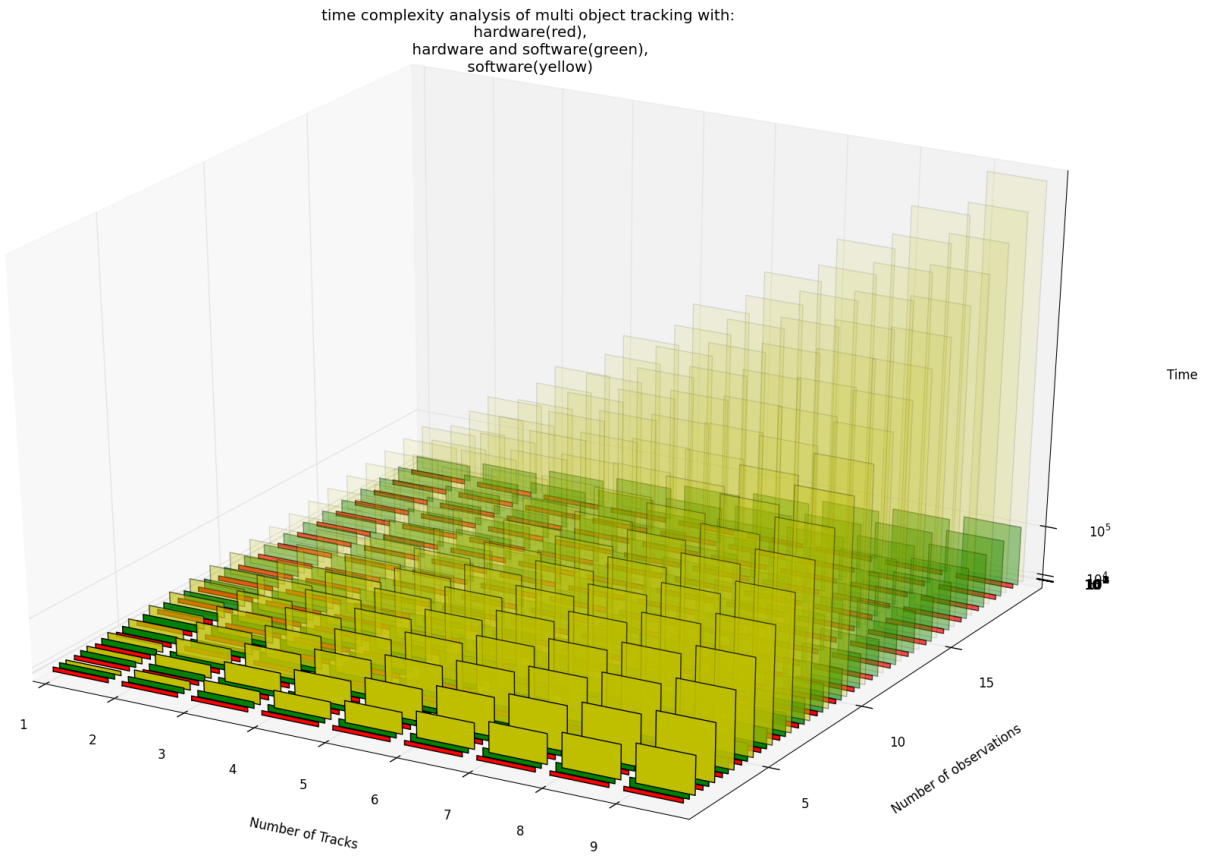


Figure 3.26: Comparison of the (formal) time complexity for tracking multiple objects with a hardware solution (red), a hardware-software solution (green), and a software solution (yellow). The hardware-software solution performs significantly better than the software solution despite the high transmission delays assumed for the calculation.

Figure 3.26. It can be seen that for a, with respect to the robotics and automotive application domain, reasonable numbers of tracks and features the hardware solution has a complexity advantage of more than 1000 in extreme cases, and of 10 to 100 in the majority of cases.

Performance Estimation for a hardware-software solution

Some of the operations needed during the normalization of the data, the conversion in the common representational format (CRF), and the calculation of the data association are algorithmically complex and can be executed very efficient in software. Fortunately some of the concepts are orthogonal and it is possible to distribute multi object particle filter to hardware and software components. This allows a faster and more convenient implementation of the algorithm and adds greater flexibility to the solution. Changes in software components and in the hardware components do not require a re-engineering of the other components as long as the interfaces are untouched, because of the orthogonality of the concepts.
The calculation of the time complexity of such a solution needs to account for the com-

munication delay that could be neglected in the time complexity calculations of the pure hardware and software solutions where communication delays where not significant. The following equation, where D_{sw}^{hw} denotes the delay during hardware to software transmissions and D_{hw}^{sw} denotes the communication delay in the other direction, estimates the time complexity of the hardware-software solution. For the calculation broadcast messages are allowed. Again c_{hw} is the time needed by each hardware particle filter to calculate the independent data association probability of the track given a feature, d_{hw} is the time needed by each hardware particle filter to update the track given a feature, $z_{hw}(M)$ is the time needed each hardware particle filter for the importance sampling, and $p_{hw}(M)$ is the time needed by each hardware particle filter to extract the probability density from the sample set of the track.

$$K_t D_{hw}^{sw} + K_t c_{hw} + K_t N_t D_{sw}^{hw} + K_t N_t a +_t D_{hw}^{sw} + K_t d_{hw} + z_{hw}(M) + p_{hw}(M) +_t D_{sw}^{hw} \quad (3.15)$$

Depending on the means of communication the delay induced by the transmission may be negligible, if for example the hardware is attached via pcie-bus. Other means of communication provide more freedom with respect to the integration of the system but may induce longer transmission delays. For example TCP/IP transmission allows the flexible local area network wide distribution of the hardware the length of the transmission delay will be in the order of milliseconds. Figure 3.26 shows the influence of the transmission delay on the time complexity of the system is not as strong as one might expect. The hardware-software solution is significantly faster than the software solution.

3.2.3 Hardware-Software Information Interfaces

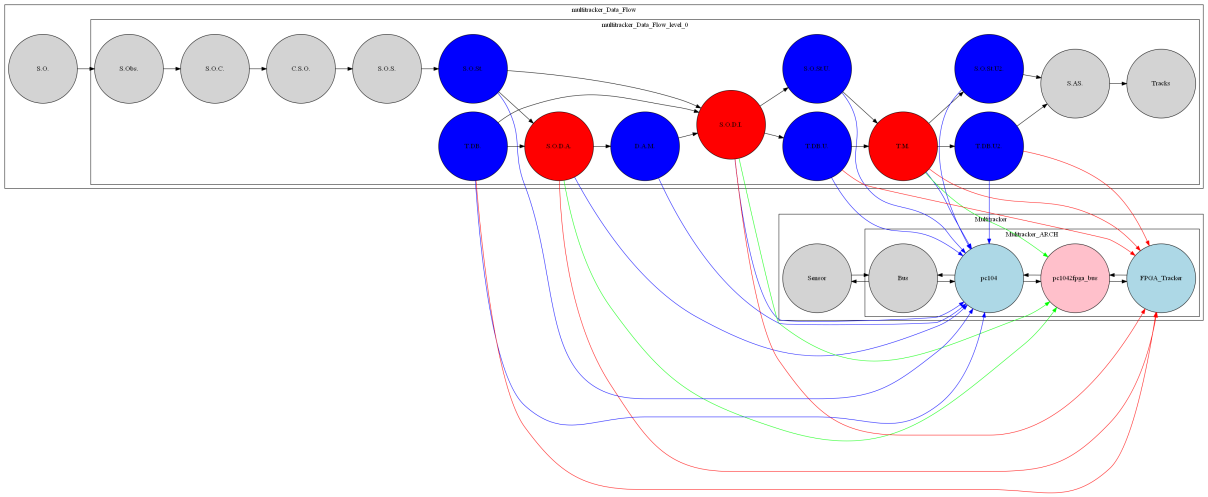


Figure 3.27: Locations of hardware-software-interfaces. Nodes in the problem graph that are bound to the pc104 and the fpga_tracker distribute their functionality and require hardware-software-interfaces to transmit data between hardware and software.

When realizing a hardware software tracking system the interfaces between hardware and software are very important. The formal interfaces were developed as soon as possible in

the stage of conceptual development, with respect to the various processes that involved the hardware and software components and the requirements and topology of the according problem graphs. The problem graph was refined during the later development process, and the interface was adopted according to the changes made during the refinement.
The initial formal interfaces were derived from the refined problem graph bound to a suitable architecture graph as in Figure 3.2.3. Since the focus is on exchange of information between hardware and software components only the formal interfaces for these parts are discussed here. With respect to the tasks needed for tracking multiple objects the individual trackers need to provide the following services:

- Initialize a new track with given parameters
- Return the independent data association probability (IDAP) of a track and an observation given an observation.
- Update an existing track given an observation a data association probability.
- Provide a compact representation of the probability density of the tracks state.
- Provide a prediction of the tracks probability density to predict track movements.
- Delete an existing track.

To keep the implementation convenient the message format was chosen in such a way that the size of the format is always the same. There are 3 types of messages, 1 from the software to the hardware and 2 from the hardware to the software. Communications are always initialized from the software side.
The communication scheme, as developed during the prototypical FPGA implementation, for the initialization of track is shown in Figure 3.28. The communications between hardware and software during the life cycle of a track are:

1. The software receives a sensor observation from a sensor.
2. The software needs to instantiate a new track and sends an unassigned observation as broadcast message with the observation to the connected hardware trackers.
3. Because the message is unassigned the trackers calculate and return the IDAP. If the tracker is not initialized it returns the value -1 to indicate its uninitialized status.
4. Once the software has received the answers of all trackers it calculates the joint data association probability and, if the observation could not be assigned to any existing track, sends the assigned observation (marked by the id of the unassigned tracker) as a broadcast message to the connected hardware trackers.
5. The uninitialized tracker reacts on the observation assigned to it and initializes the track with parameters based on the observation.
6. The following steps a repeated until the tracker does not receive any more observations originating from the tracked object.
   (a) The software receives a sensor observation from a sensor.

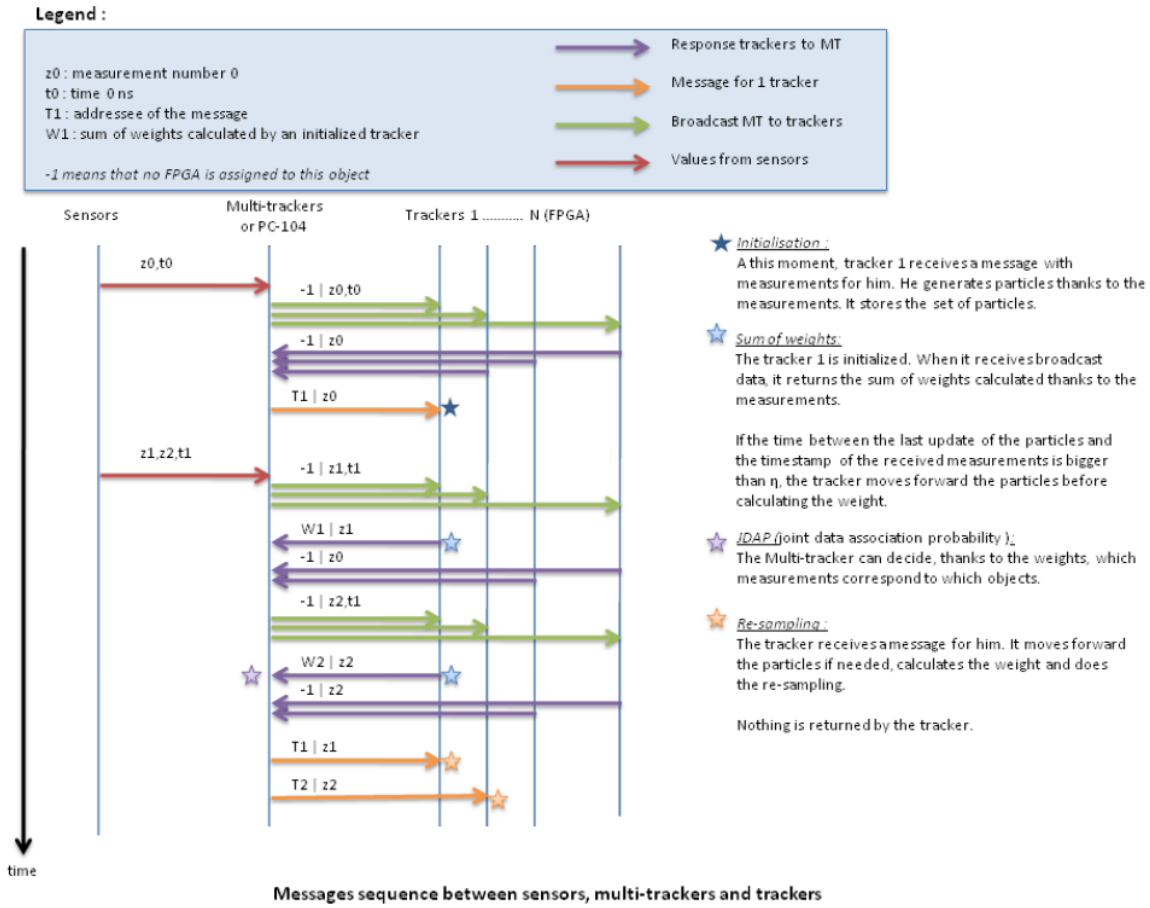


Figure 3.28: Communications during the initialization of a track. The chart is from the report on the initial development of the prototype [14].

(b) The software sends an unassigned observation as broadcast message with the observation to the connected hardware trackers.

(c) Trackers calculate and return the IDAP.

(d) Software calculates JDAP from the received IDAPs.

(e) Software sends assigned observations to trackers.

(f) Trackers integrate assigned observations and send back a normal distribution approximation of the tracks probability distribution.

7. There where no observations assigned to the track for such a long time that it is most likely that the object has left the area, the software broadcasts a message with the id of the track to remove the track.

8. The hardware with the according id returns to an uninitialized state.

To support these operations the messages were defined in the following way.

Software to hardware: The following data is sufficient, when using a clever communication scheme and overloading data fields, to do all the necessary communications with the hardware needed to realize the operations as defined above.

- The Joint data association probability, values outside the interval [0,1] are used for control messages.
- The id of the tracker (receiver) (0 = broadcast), and the id of the track.
- The measurement header consisting of: The id of the sensor that made the observation, the time-stamp of the measurement with seconds and microseconds, the number of the measurement, and the series of the measurement.
- The mean of the measurement.
- The covariance of the measurement.
- The determinant of the covariance matrix.
- The inverse of the covariance matrix.
- The TAG matrix.

Hardware to software: There are two types of messages send from the hardware to the software. One is used for the intermediate calculation of the joint data association probability and the other to report the state of a tracked object.

- The independent data association probability (-1 if not initialized, else sum of weights).
- The id of the tracker (receiver) and the id of the track (negative if no track assigned).
- The measurement header consisting of: The id of the sensor that made the observation, the time-stamp of the measurement with seconds and microseconds, the number of the measurement, and the series of the measurement.

The tracker must also provide an normal probability density function approximating the state of the tracked object so the software or other superordinate instances can use the information. In addition to the probability density function the probability that the track really exists is returned and also the data needed to map the report to the according object and time.

- The probability that the track exists. This is important for the track management that relies on this information to decide which track are to be uninitialized.
- The id of the tracker (receiver) and the id of the track (negative if no track assigned).
- A time-stamp including seconds and microseconds that determines the time of the tracks state estimate in the common time line.
- A normal distribution approximation of the tracks state.

3.3 Inclusion of Domain Specific Information

A requirement for a general embedded tracking system is that it is easily adoptable to different application scenarios, and that it can be integrated with minimal support into existing systems. The inclusion of domain specific or application scenario specific background information allows the adoption of the general tracking process to specific tracking problems, and it can enhance the performance of the tracking significantly.

3.3.1 Dynamic Sensor Uncertainty Maps

The uncertainty of sensors is often not fixed for a given tracking scenario, but it changes depending on the location of the measurement, the object measured, and environmental conditions that influence the measurement. The following example will explain this effect. In recent studies on the performance of multilateral (MLAT) position detection [78, 77] it was found that the accuracy of the measurements decreased as more aircraft were in the area of the measurement. This may be a possible result of multi path reflections from the surfaces of the aircraft. Also the uncertainty is depended on the location of the measurement as the uncertainty plot in Figure 3.29 illustrates. Because of the positioning of the MLAT transceivers and the location of buildings etc. the uncertainty varies at different parts of the runway entrance. The plot is based on the original data from the Narita airport provided by the authors of the above studies. If the tracking does not consider the local uncertainty the results of the tracking are inaccurate and may even be misleading.
Another example is from the robotics domain. The authors of a study [74] show that the uncertainty of measurements used for the localization of a mobile robot are depended on the position of the robot in its environment. They improved the quality of the localization by integrating the local uncertainty into the tracking algorithm.
In order to integrate the local uncertainty the hardware needs to support dynamic uncertainty adjustments of the uncertainty of connected sensors. Because the integration of the representation of the uncertainty map is quite complex under the aspect of concurrency, the hardware only supports individual uncertainty per measurement, which is sufficient to allow the insertion of the sensors measurements local uncertainty by software.

3.3.2 Object State Transition Dynamics

Besides the uncertainty of measurements one of the most domain dependent aspects of tracking is the behavior of tracked objects. The behavior of cars depends strongly on the road topology, the traffic situation, and other circumstances. Drivers will hardly behave the same on a parking lot, in city traffic, or at the highway not to mention influences such as rush hour or traffic jam. Also the physical characteristics of the traffic participants influence the behavior. A sports car or a motor cycle will accelerate much faster than a truck.
Robots interacting with humans and autonomously navigating in domestic environments have to ensure that their movement is no risk to humans. While it might be sufficient to employ reactive control to avoid collisions with pedestrians in a shopping mall, a robot moving on the sidewalk will not be able to avoid collisions without a reasonable good estimate of the movements of the people on the sideways.

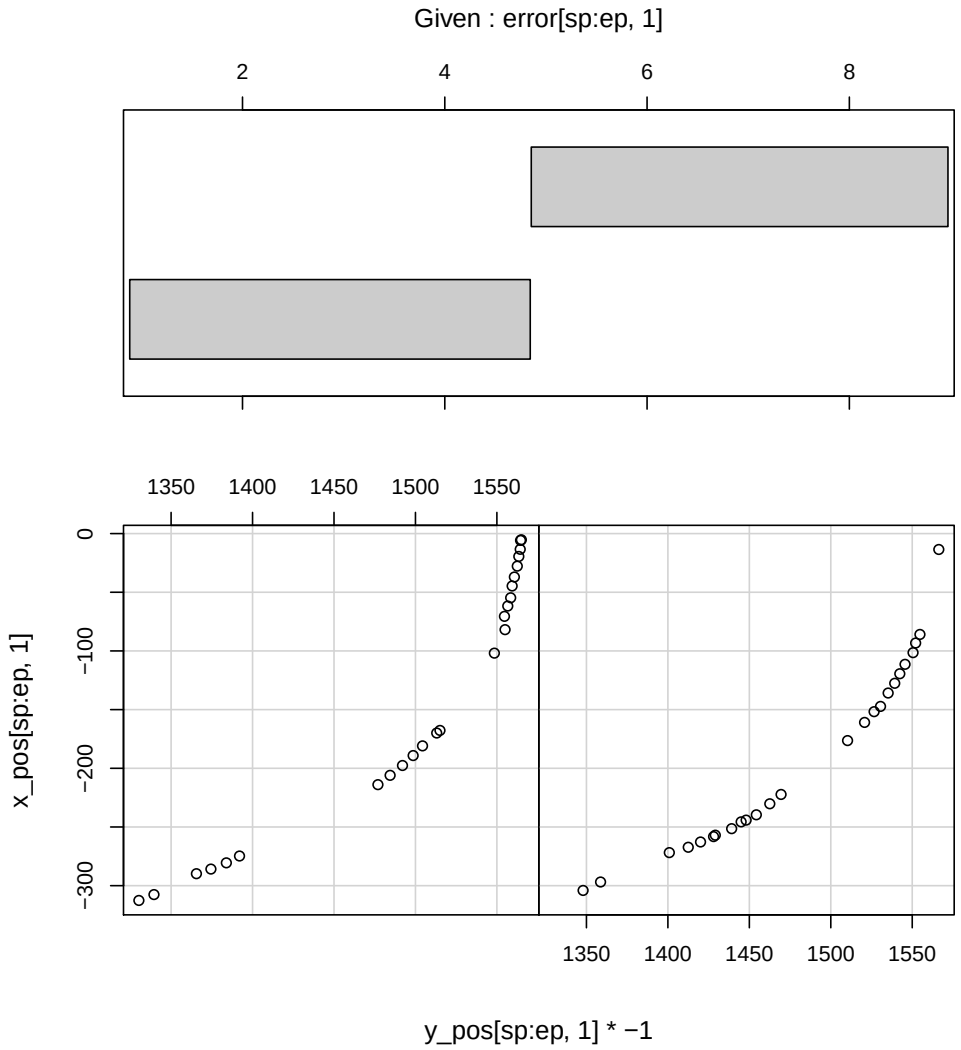


Figure 3.29: Local uncertainty values of MLAT measurements. The plot shows that the uncertainty of the measurements increase and decrease in correlation with the location of the vehicle as it travels along the taxiway.

An interesting application of domain specific dynamics is the derivation of the semantics of the environment based on the behavior of the tracks. Particle filter can be easily extended to support domain information by adding a domain field to the probability density function space, see [100]. The filter then estimates the likelihood of the domain or a specific behavior/maneuver based on the movement of the vehicle.

Vehicle and Domain specific State Transitions

For these reasons it is recommendable to use vehicle specific state transition matrices that capture the usual behavior of a vehicle with respect to its physical constraints and the environment. The most simple support is the hardware support of the update of the tracked objects state transition matrix. A state transition matrix is description of the parameters of a state transition function that is implemented in the hardware. By

changing the matrix, and therefore the parameters, the state transition of the object can be influenced according to the object type and or domain.

Implications for the formal system architecture

The inclusion of domain specific context in the tracking system requires that the hardware's algorithms make use of methods that can be tuned to model the circumstances of specific domains by a set of parameters, which affect the mathematical calculations and algorithmic behavior of the hardware based tracking of individual objects. Therefore, in order to support the incorporation of domain specific information, the formal interface should include the data necessary to update according models on the hardware side during run time. The details on the realization of the domain specific support are explained in Chapter 4.

3.4 Summary

This chapter applied the theories, methods, and insights of the previous chapter on embedding multi modal sensor data fusion to the problem of efficiently tracking multiple objects using hardware and software. The basic formal models, algorithms, architecture, partitioning, and interfaces for the system were introduced and motivated. While the rest of the task to be done seems to be an engineering problem, it will quickly become apparent that this is not the case. As always with engineering problems there are plenty opportunities to do scientific work and find a new, better way to deal with problems than the usual approach allows.

Chapter 4

FPGA Particle Filter

Contents

4.1 Introducion

A particle filter is a non parametric implementation of the Bayes Filter. For tracking and localization in the robotic domain particle filters have a lot of interesting properties, and many works have been published on the topic of tracking and data fusion with particle filters [25] [39] [47] [96] [85]. The most interesting property is the ability to approximate any parametric representation of a distribution using a finite set of samples. This is very important in robotic applications where tracked objects often do not exhibit a linear behavior. Also sensor measurements and detected features might have different kinds of parametric uncertainty distributions, which makes it difficult to apply them to single parametric probability distribution like a Gaussian distribution in a Kalman filter used for system state estimation. Nevertheless particle filters can integrate information presented as closed form parametric probability distributions. This flexibility allows a very efficient implementation architecture of a distributed tracking system.

4.1.1 State of the Art

There are numerous works on particle filters and their mathematical and algorithmic properties. The accuracy and error of particle filters is an active field of research. The performance of particle filters can be estimated in comparison to the optimal filter. The optimal filter is, except for a very restricted set of dynamic models not expressible in a closed form. It has been shown that the upper bound on the variance of estimation error depends on the number of samples M, and has the form $c \cdot O(M^{-1})$ that is independent to the number of dimensions of the sample state space, but it has also been shown that the constant c heavily depends on the state vector dimension [21, 19]. The grow of c is linear to the number of dimensions with a good importance density respectively proposal distribution; with a bad proposal distribution the variance of error becomes exponential to the dimension of the state vector. This "curse of dimensionality" has been experienced in many practical applications [91] and also applies to robotic applications like tracking or pose estimation. While tracking stationary objects on a 2 dimensional map requires only 3 dimensional state vector, tracking moving objects in a 3 dimensional volume requires a 12 dimensional state vector. Therefore a high number of samples is necessary to keep the error small and enable the robot to achieve its goals when dealing with complex environments like domestic surroundings. FPGAs have been used to implement particle filters [118, 45] for various application areas, and it has been shown that some parts of the algorithm can be executed concurrently, but i have not found publications on the implementation specialized for small autonomous robots.

4.1.2 The Concurrent PFA

This subsection describes the conversion of the sequential particle filter algorithm Particle Filter (X_{t-1}, u_t, z_t) to an algorithm that supports partial concurrent execution.

Algorithm Particle Filter (X_{t-1}, u_t, z_t)**:**

```
1 X̄_t = X_t = ∅;
2 for m = 1 to M do
3 |   sample x_t^[m] ~ p(x_t|u_t, x_{t-1}^[m]);
4 |   w_t^[m] = p(z_t|x_t^[m]);
5 |   X̄_t = X̄_t + (x_t^[m], w_t^[m]);
6 for m = 1 to M do
7 |   draw i with probability ∝ w_t^[m];
8 |   add x_t^[i] to X_t;
9 return X_t;
```

The idea of a particle filter is to represent the probability distribution of a system state at time t, with a finite set of samples X_t, where the density of the samples approximates the probability distribution. In a particle filter the samples are called particles and are usually denoted as $x_t \in X_t$. Each particle is a hypothesis to what the true state of the system may be at time t. Attached to each sample x_t is a weight w_t. The weight is important because it is used to approximate the target probability distribution, using the technique of importance sampling. The core of the algorithm of the particle filter is shown in the Algorithm Particle Filter (X_{t-1}, u_t, z_t) and works as follows: The particle filter takes a sample set X_{t-1}, information about the transition probability u_t, the observations z_t and returns a new sample set X_t approximating the probability density function of the current system state. The first part of the algorithm is analog to the Bayes filter, in line 3 the transition function is applied and then, in line 4, the observation is applied by calculating the weight of the sample. In lines 6 through 8 the algorithm draws with replacement M particles from a temporary set $\overline{X_t}$. The probability of drawing a particle is given by its importance weight. This technique is known as "resampling " or "importance sampling". The interesting part, with respect to a concurrent implementation, of the filter are the "for" loops. The first loop ,lines 2 - 5, and the second loop, lines 6 - 8, are well suited for a single instruction multiple data (SIMD) computer implementation. However the standard architecture models for SIMD computers , the arrayprocessor and the vectorprocessor, cannot be applied directly to the problem due to the probabilistic nature of the algorithm that raises some issues requiring special treatment which affects the architecture.

Concurrency

The timing of the serial computation of the PFA is similar to the one shown in Figure 4.1, where a single general purpose processor (GPP) is used to execute the operations of the PFA. While the GPP is seldom idle, the time needed to execute the PFA can, depending on the number of samples, become quite long.
Basically a custom or semi-custom PLD implementation offers the freedom of implementing multiple single purpose processors (SPP) allowing the concurrent execution of the different operations in the particle filter algorithm. However due to the nature of

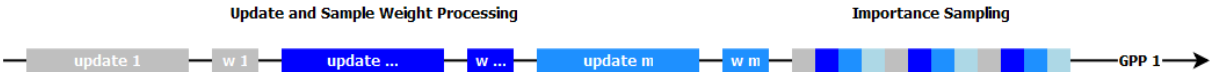


Figure 4.1: Timing of the serial execution of the PFA by a General Purpose Processor.

the mathematical model the parallel implementation of the resampling has been only partly successful [87]. The resulting concurrent algorithm is listed in Algorithm Concurrent Particle Filter (X_{t-1}, u_t, z_t). The primary difference is the introduction of the variable K that represents the number of SPP used for the sample update and the calculation of the sample weight. Thus the CPFA algorithm iterates the first loop for $\frac{M}{K}$ times and does the update and sample weight calculation for K samples in parallel.

Algorithm Concurrent Particle Filter (X_{t-1}, u_t, z_t):

```
1  X̄_t = X_t = ∅;
2  for m = 1 to M/K do
3      for m = 1 to K concurrently do
4          sample x_t^[m+k] ~ p(x_t | u_t, x_{t-1}^[m+k]);
5          w_t^[m+k] = p(z_t | x_t^[m+tk]);
6          X̄_t = X̄_t + (x_t^[m+k], w_t^[m+k]);
7  for m = 1 to M do
8      draw i with probability ∝ w_t^[m];
9      add x_t^[i] to X_t;
10 return X_t;
```

The second loop is often kept serial for a number of reasons. The most important is that, because of the nature of the underlying mathematical model, the importance sampling must use the normalized sample weight $w_t^{[m]}$ that is only accessible after all sample weights have been calculated. Therefore there is no published solution to fully integrate the importance sampling in the update-sample-weight-loop. Another reason is that the overhead for coordination of the importance sampling grows with K and requires more resources. Unless there are as many processing units (PUs) as samples , so $K = M$, intermediate storage is required for the samples. In this case samples are stored in RAM which usually does not support to write data parallel to multiple addresses, which would effectively serialize a parallel importance sampling. On the other hand, if $K = M$ then no intermediate storage is required as all samples data is kept in the PUs. The problem that arises here is to distribute the sample data from one PU to another PU for M PUs concurrently. While this is theoretically possible for arbitrary sized sample sets, the resources required for reasonable sized sample set are impractical, because the required bus needs a bandwidth of $(M-1)^2 \cdot size(sample)$ bit per clock, e.g. A sample set with 500 samples represented by 224 bit requires app. 6800 MB per clk bandwidth, a sample set with 128 samples still requires app. 440 MB per clk bandwidth. Thus resampling is mostly done serial and some of the computational units remain idle during the execution

of the PFL for a part of the time, see Figure 4.2.

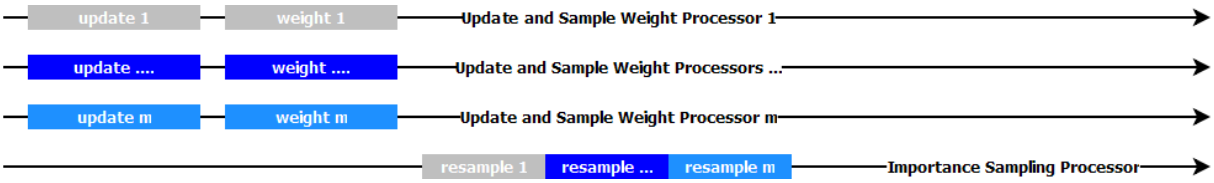


Figure 4.2: Timing of the parallel execution of the update and sample weight calculation with a serial resampling.

Another factor that contributes to the time in which computational resources are not used is the rate of observations compared to the speed of the concurrent particle filter algorithm (CPFA). The frequency of sensor observations z_t that need to be integrated in the tracking is, with respect to the speed of the most parallel implementation of the PFA, comparatively low, but the large number of SPP requires a lot of chip area. Against this background the pipe lining of the algorithm offers a lot of advantages. Figure 4.3 shows the timing of a partially pipelined execution of the algorithm where 4 SPP's dedicated to the update procedure perform the upgrade of 4 samples concurrently and the resulting samples are processed by another SPP dedicated to the calculation of the sample weight. After the completion of the partly parallel pipeline the resampling is done by a SPP dedicated to the importance sampling.

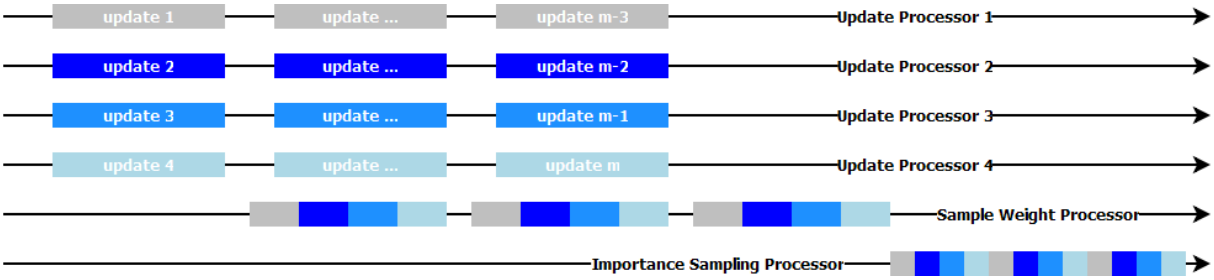


Figure 4.3: Timeschedule of the particle filter algorithm in a concurrent, pipelined architecture.

The most important variables that govern the efficient realization of the overall architecture are the total execution time and the total chip area needed. In order to achieve an optimal implementation one needs to consider the nature of the individual operations of the algorithm with respect to the chosen mathematical models. All mathematical models can be broken down into the most basic operations addition, subtraction, division, and multiplication. The mathematical models that have been introduced in the previous chapter rely on vector and matrix multiplications for booth sample update and sample weight calculation. Thus, while the implementation of these steps requires separate finite state machines for the control, arithmetic, and logic units can be shared.

4.1.3 FPGA Number Representation

In order to achieve a maximum performance and accuracy many samples are needed, the more the better. For application in small robots logic cell usage, power consumption and heat dissipation must be kept at a reasonable level.

When dealing with the design of massive parallel computation hardware the format chosen for the representation of numbers is an important design decision. One of the more important factors that guide the decision are the dynamic range and resolution of the format, with respect to range and precision needed in the computation of the algorithms in front of the background of the application context. This affects not only the range of the input and output values of the system, intermediate calculations also need to be considered. These are discussed in detail in the following Section 4.2. Floating point representations offer a wider dynamic range and dynamic precision, but the results of mathematical operations like multiplication, division, addition, or subtraction are not necessarily sound within the dynamic range. Fixed point representations do not suffer from arithmetic errors during addition and subtraction if the result remains within the range of the representation, but the dynamic range is considerably low compared to floating point representations when using the same number of bits for both formats.
Another point are the cost of the mathematical operations in time, chip area, and power consumption. Research that performed comparative studies found that on modern FPGA devices the cost for fixed point addition and subtraction are significantly lower in the above mentioned terms. The cost for multiplication and division of fixed point numbers are moderately lower compared to the cost of the same operations on floating point numbers [37]. Here the time needed for floating point multiplications can be reduced significantly using pipe lining, but at the expense of chip area. According to the results published in [37] addition of a 32-bit floating point number requires app. 14 times more chip area than the addition of 32-bit fixed point numbers and requires app. 6-7 times more energy. Results for multiplication show that a 32-bit floating point multiplication requires app 2,5 times more chip area and app 1,3 times more power than a 32-bit fixed point implementation. These results are shown in a comparative chart in Figure 4.4.

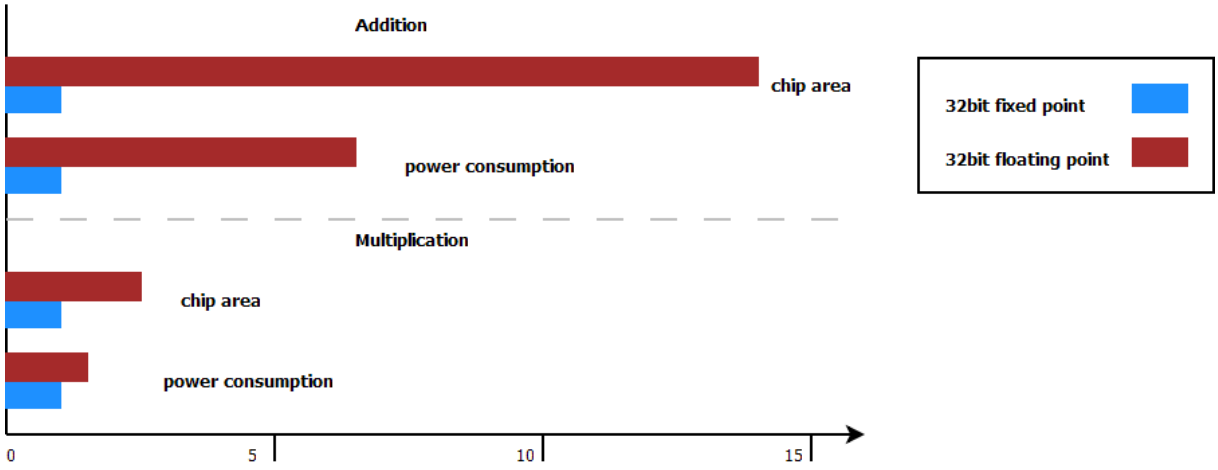


Figure 4.4: Normalized comparative chart of the power consumption and chip area needed for the addition and multiplication of 32-bit fixed and floating point numbers.

The larger amount of time, chip area, and power needed is mostly a result of the computational overhead required for the floating point operations, particularly the normalization. This overhead can be reduced significantly in pipelined complex operations like matrix multiply, where serial aspects inherent to the mathematical algorithm do not allow for a full parallelism. Therefore some scientist suggest using a mix of fixed and floating point

numbers in complex calculations [22]. Given the properties of the concurrent particle filter algorithm and the application context chip area, power consumption, and heat dissipation are significant design constraints. The advantage achieved through the usage of fixed point representations was estimated in between 5-10 times more processing units when using fixed point representations while consuming about only as much as half of the power at the beginning of the development. The nature of the application scenario and the mathematical models involved in the necessary calculations also support the usage of fixed point numbers. Thus fixed point number representations have been chosen for the FPGA implementation of the particle filter algorithm. For most of the calculations 32 bit values are used. The interpretation of the numbers depends on the usage, so positions are interpreted as 22 bit integer and 10 bit fractional part to cover a medium area with reasonable accuracy. For probabilities almost all bits represent the fractional part. The fixed point number representation of the numbers involved in the computation intensify some problems encountered with digital implementations of particle filters and require a special treatment and an adaption of the particle filter algorithm. As the focus of the research is on the mathematical and algorithmic aspects of the concurrent implementation the value size was not optimized in the prototypical implementation due to time constraints, although this would have led to a better performance as shown later in this chapter.

4.2 Implementation Architecture

This section describes the architecture of the FPGA particle filter (FPF), based on the previous discussion of the concurrent particle filter algorithm and the formal implementation architecture. The section begins with a description of the architecture and the components and continues mapping the architectures components to the serial PFA and concurrent PFA, providing an in depth description of the different parts of the hardware that implement the different parts of the algorithms.

4.2.1 Architecture Overview

The Architecture has two main components: The first component is the NIOS Softcore Processor with peripheral interfaces, the second component is the particle filter logic (PFL). The consists of the following VHDL components, the actual layout in on the FPGA is shown in Figure 4.5:

- pf_top: The top level component that controls all of the other components and is responsible for the execution of the PFA.
- lfsr_generic: random_component: Generates random numbers. Random numbers are needed in various stages of the particle filter algorithm.
- sample_weight_processor: SWP_inst: Computes sample weights. This unit computes the weight w of a given sample with respect to a given sensor observation.
- bus_manager: Manages the routing of data between the components of the PFL according on the state of the PFL. This enables the usage of the same data lines for

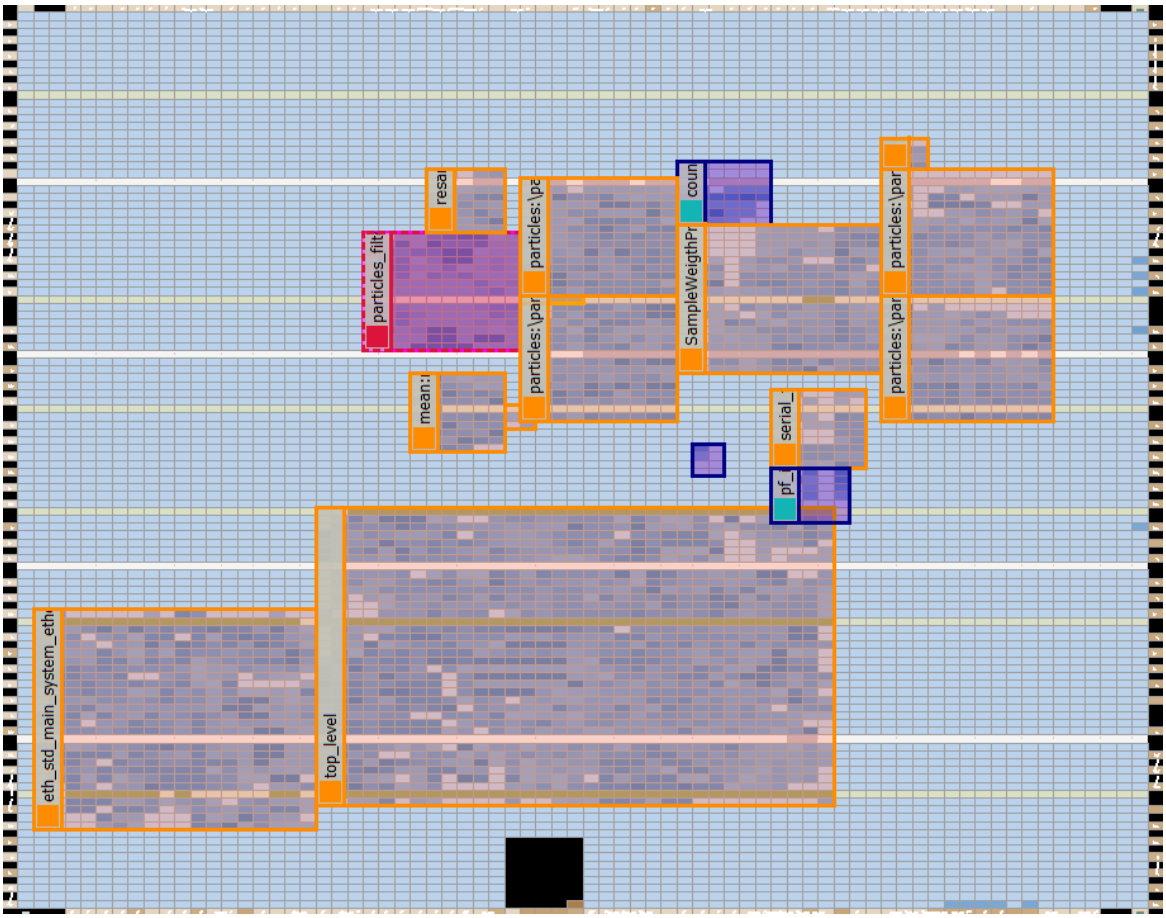


Figure 4.5: A possible layout of the components in the chip planner.

different purposes during the various stages of the CPFA, which results in a smaller chip area required for the interconnecting signals.

- count_particles: This instance is required for the application in a special airport traffic surveillance scenario. It counts the samples on either side of a virtual line.
- mean: Computes the - 6 dimensional - mean of the sample set. This functionality is required to communicate the probability distribution of the samples in a compact for to superordinate instances.
- particles0-3: particles_inst: These components perform the update and initialization steps of the CPFA.
- particles_loader: This component is responsible for loading samples from the sample storage and saving updated samples in the storage.
- resample: Implements the resampling step of the particle filter algorithm.
- serial_to_mem: Stores data received by the serial interface to the PFL memory and fetches data requested by the serial interface from the PFL memory.

- single_port_ram: The PFL internal memory (sample storage) where the sample set is stored.

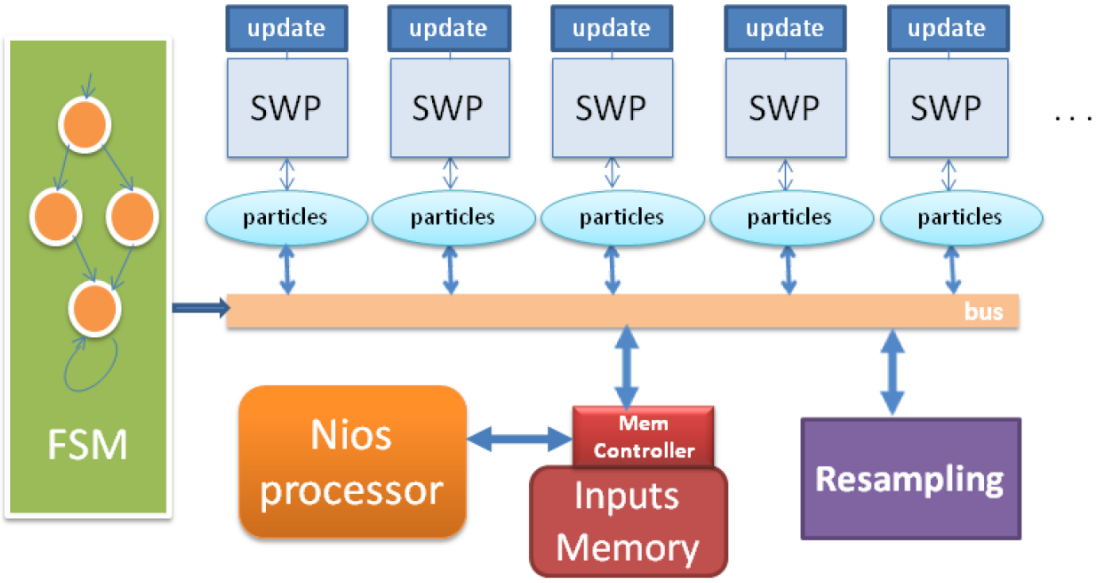


Figure 4.6: Architecture of the FPGA particle filter implementation.

4.2.2 Representation of Samples

Each sample $x_t^{[m]}$ in the sample set X_t represents a point in the state space of the system, in this case a mobile object. Based on the results during the initial feasibility study [100] a subset of the state space variables that is sufficient to model movements on a flat map has been chosen for the prototype implementation. Previous work has shown that errors induced by barrel projection of airports the size of the Hamburg Airport do not negatively affect the tracking [100]. A 3 dimensional representation would require a 12 dimensional sample state space, which is unfavorable due to the properties of the PFA discusses earlier in this section. The subset is mathematically a six element vector $x_t^{[m]} = (t_x, t_y, \theta, \dot{t}_x, \dot{t}_y, \dot{\theta})$ that includes:

1. 2D-Translation: t_x and t_y encode the position of the vehicle on a flat map.
2. Heading: The orientation of the vehicle on the map is stored in θ.
3. 2D-Translation Speed: $\dot{t}_x$ and $\dot{t}_y$ encode the change in translation over time of the vehicle.
4. Rotation Speed: The change in orientation over time is given by $\dot{\theta}$.

Each value of $x_t^{[m]}$ is stored as a 32-bit number, semantically interpreted as fixed point, where 10 bits are used for the fractional part and 22 bits are used for the integer part and sign. On a meter scale this results in a precision of app. 1 millimeter and a map with a range of +-2097 km in x and y direction.

Representation of position and speed

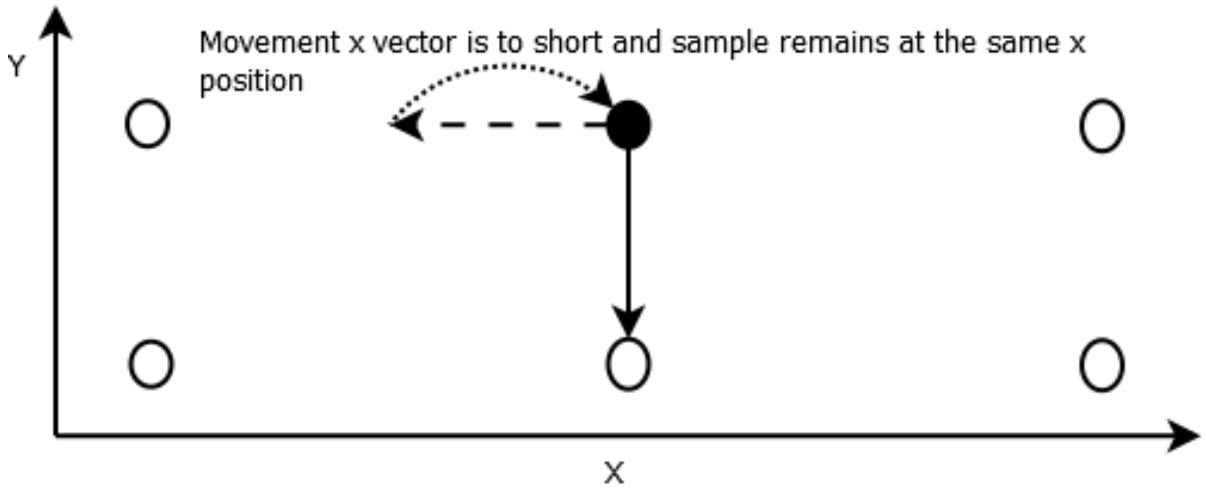


Figure 4.7: The speed-position precision problem. The circles mark the discrete positions representable by a sample. If the translation speed of a sample is to slow - with respect to the given update rate and the sample position precision - the sample remains at its position.

This uniform 32-bit representation, which is used in the prototype for the sake of a timely implementation, is not really needed for many application scenarios. For most application scenarios the samples do not spread across a wide area and only a limited precision is needed. For most applications a reasonable area of surveillance is so big that a target of interest speed limit enables multiple measurements of the target. Therefore the speed $\dot{x}, \dot{y}$ requires about 3 bit less for the integer part compared to the integer part of the position x, y. The size of the fractional part of the speed and the position depends on the accuracy needed in the application scenario and on the movement update interval of the samples. If the samples speed is very low and the update interval is high, then the movement of the target is cut away and the samples stick to their position if the fractional part is to small. As a rule of thumb, most application scenarios track mobiles that are either moving with a reasonable speed or not moving at all. A mobile moving very slow will produce many observations and the movement update can be neglected, because in such a case the sample set converges on the true position by replacing samples stuck to the old position with samples stuck to the new position. Therefore the fractional part of the speed should be chosen such that the reasonable minimal speed of moving mobiles can be represented[1]. The fractional part of the position needs to be chosen such that the samples can move from their current position to the next given the fractional part of the speed and the update rate during the tracking[2]. Figure 4.7 illustrates this fact. Another requirement is that the precision of the position must be chosen in such a way that the main moment of the probability density function of a sensor measurement spreads across multiple discrete position points. Also the heading θ requires only a signed 3 bit integer to cover the interval [-pi, pi] and 12 bit fractional part should be enough for most tracking

[1]E.g.: Airport Ground Traffic surveillance with meter-decimeter per second.
[2]E.g.: Decimeter-centimeter for Airport ground traffic surveillance.

scenarios [3]. Except for tumbling target scenarios the rotation speed $\dot{\theta}$ is expected to reside within the same range as the heading θ.

Representation of the sample weights

Attached to each sample $x_t^{[m]}$ is a weight $w_t^{[m]}$ that represents the quality of the sample or probability of the state of the sample given the course of observations of the vehicle; formally: The weight w_t of a sample x_t in a sample set X_t is determined by the probability $p(z_t|x_t)$ of a sensor observation z_t given x_t. Due to the discrete sampling point nature of the particle $p(x_t)$ is 1 and $p(z_t|x_t) = z_t(x_t)$[4]. In the prototype implementation the default interpretation using 22 bit integer and 10 bit fractional part hast been chosen for the reason of timely implementation. This is not optimal because a, with respect to the optimal usage of the fixed point representation, reasonable chosen probability density function (PDF) should have no value equal to or bigger than 1 at any point, and the fractional part can be reduced appropriate to the PDFs of the sensors in the application scenario. The fractional part must be chosen in such a way that the resulting discretized PDF has a main moment that is a stair and not a plateau, which is required to ensure that the particle filter converges to the true posterior.

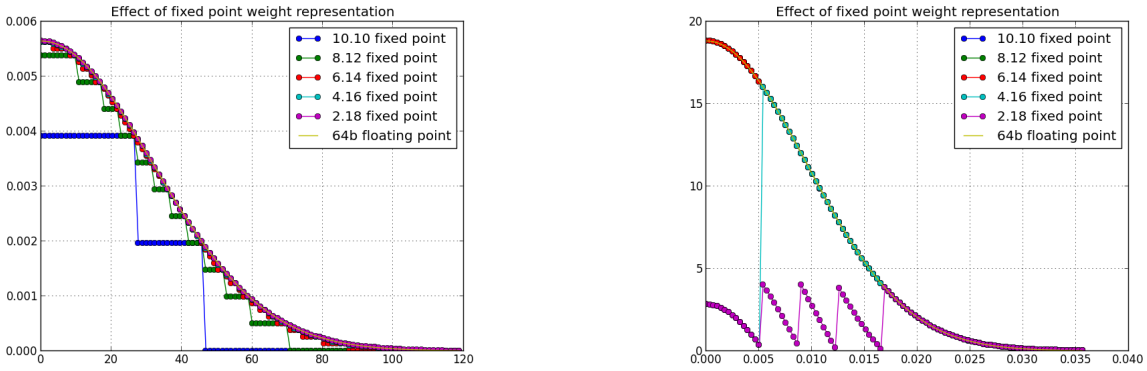


Figure 4.8: Influence of the fixed point representation of the weight of the sample on sensor measurement probability density function discretization. Depending on the unit of measurement and the uncertainty of the sensor the different representation are not equally suited to represent the probability.

Figure 4.8 illustrates the difference between the weight w_t of a sample x_t in a sample set X_t and the real probability of the system state represented by the sample for a selection of fixed point number representations. Depending on the distribution of the samples across the systems state space, the probabilities of the samples given an observation $p(z_t|x_t)$ can be very different, and it happens that the weight is not equal to the probability of the sample given the observation, so $w_i \neq p(z_t|x_t)$. If for example $p = 2^{-34}$ and only 32

[3]Most sensors provide a much smaller resolution when detecting the heading of an object.

[4]This requires a normalization to ensure that the sum of sample weights satisfies the law of total probability, and treating X_t as a delta distribution. The simplification is not sound for in general, but only for specific models and algorithms like the particle filter and only under constrained circumstances.

digits are available for the fixed point representation, then the weight is zero - $w_t = 0$ - because fractional bits representing the number are cut off. This can result in a loss of information that can be considered a systematic error. Furthermore a sample weight of 0 is an important case of error, because it can cause arithmetic errors during the normalization of the sample weights. On the contrary a too limited scale for the state space of the sample can result in an unreasonable high probability not representable by the fixed point number, $w_t > 2^{maxbit}$. Both phenomenons require a special treatment to ensure optimal accuracy and convergence of the particle filter.

Table 4.1: Comparison of the sample representation for different application scenarios.

Scenario	Value	integer	fractional	range	precision
Prototype	x	22	10	+-2097 km	millimeter
Prototype	y	22	10	+-2097 km	millimeter
Prototype	θ	22	10	+- $667k\pi$	microradians
Prototype	$\dot{x}$	22	10	+-2097 km/s	millimeter/second
Prototype	$\dot{y}$	22	10	+-2097 m/s	millimeter/second
Prototype	$\dot{\theta}$	22	10	+-$667k\pi$	microradians/second
Prototype	weight	22	10	2097k	1/1024
Prototype	sum	156	70	4497409 km^2	millimeter
AGTS	x	12	7	+-2048 meter	centimeter
AGTS	y	12	7	+-2048 meter	centimeter
AGTS	θ	3	12	+- pi	microradians
AGTS	$\dot{x}$	9	4	+-256 m/s	decimeter/second
AGTS	$\dot{y}$	9	4	+-256 m/s	decimeter/second
AGTS	$\dot{\theta}$	3	12	+-π	microradians/second
AGTS	weight	0	16	1	1/65536
AGTS	sum	48	62	4 km^2	centimeter
RoboCup	x	5	10	+-16 meter	millmeter
RoboCup	y	5	10	+-16 meter	millmeter
RoboCup	θ	3	8	+- pi	milliradians
RoboCup	$\dot{x}$	4	7	+-8 m/s	centimeter/second
RoboCup	$\dot{y}$	4	7	+-8 m/s	centimeter/second
RoboCup	$\dot{\theta}$	3	8	+-π	milliradians/second
RoboCup	weight	0	12	1	1/4096
RoboCup	sum	24	62	1024 m^2	millimeter

4.2.3 State Transition Update

The state transition update (STU), shown as boxes labeled "update" in Figure 4.6, computes the line 3 of the PFA (see Algorithm Particle Filter (X_{t-1}, u_t, z_t)):
At this point of the PFA the sample state is updated based on the previous sample state and the state transition probability of the sample state over time. The state transition update is done in a dedicated hardware entity, because it's mathematical model and it's implementation is, except for the sample representation, a concept orthogonal to the rest of the concepts of the particle filter. The update only depends on the information contained in the sample and the time difference between the last sample update time and

Algorithm Particle Filter (X_{t-1}, u_t, z_t):

```
1 X̄_t = X_t = ∅;
2 for m = 1 to M do
3     sample x_t^[m] ~ p(x_t | u_t, x_{t-1}^[m]);
4     w_t^[m] = p(z_t | x_t^[m]);
5     X̄_t = X̄_t + (x_t^[m], w_t^[m]);
6 for m = 1 to M do
7     draw i with probability ∝ w_t^[m];
8     add x_t^[i] to X_t;
9 return X_t;
```

the current sample update time. The update procedure may be changed and auxiliary and background knowledge may be included without the need to change other components of the particle filter logic.
In the prototype the update is realized as a matrix operation. Here the vector $x_{t-1}^{[m]}$, which is given by state represented by the mth sample, is multiplied with the state transition matrix $A_{\Delta t}$ that encodes how each of the state variables evolves and affect each other depending on the time difference $_{\Delta t}$.

$$x_t^{[m]} = x_{t-1}^{[m]} \cdot A_{\Delta t} \tag{4.1}$$

The equation with the state transition matrix for a plane might look, given that $t_x, t_y, \dot{t}_x, \dot{t}_y$, are in meter per second and $\theta, \dot{\theta}$ is in radians per second, like the following equation.

$$\begin{pmatrix} t_x \\ t_y \\ \theta \\ \dot{t}_x \\ \dot{t}_y \\ \dot{\theta} \end{pmatrix}_t = \begin{pmatrix} t_x \\ t_y \\ \theta \\ \dot{t}_x \\ \dot{t}_y \\ \dot{\theta} \end{pmatrix}_{t-1} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \Delta_t & 0 & 0 & 1 & 0 & 0 \\ 0 & \Delta_t & 0 & 0 & 1 & 0 \\ 0 & 0 & \Delta_t & 0 & 0 & 1 \end{pmatrix} \tag{4.2}$$

Here Δ_t is the time difference in seconds. The state transition matrix A can be read from the bus concurrently by each STU, which allows to switch or modify state transition matrices by software as needed. The orientation θ requires special treatment during the extraction of the probability density function, because the values must be normalized into the interval $[-\pi, \pi)$ for a sound representation. Usually the PSA requires the addition of some Gaussian noise to the samples to prevent the duplication of particles. In favor of computational resources this is not implemented but rather a part of the sample set is replaced by samples generated from sensor measurements. The details are explained in the following text.

Process Noise Addition Workaround

The original particle filter assumes the addition of unbiased normal distributed noise during the sample update. There are two main reasons for this:

1. Normal distributed process noise: For many application scenarios the process model is influenced by Gaussian distributed noise. The noise results from a large number of random distributed microscopic influences on the process that sum up to a Gaussian

distributed influence on the process. A good example is a car driving on a straight road; the constitution of the road and the car force the route of the car to slowly deviate from the roads center line. Therefore an appropriate mathematical model should capture these aspects of the system.

2. Sample deprivation: If the samples are not varied the importance sampling leads to a duplication of the samples with the highest weights. If no noise is added to the values of the samples, the variance of the values decreases until all samples share the same value.

A possible solution to overcome both of this problems is to replace a random subset of the sample set with samples generated from sensor observations. The samples are generated by adding equally distributed noise to the mean of the observation. This increases the variance of the sample set, and at the same time it counters the slower rate of convergence that is a side effect of the alternative zero weight handling discussed later in this section.

4.2.4 Sample Weight Processor

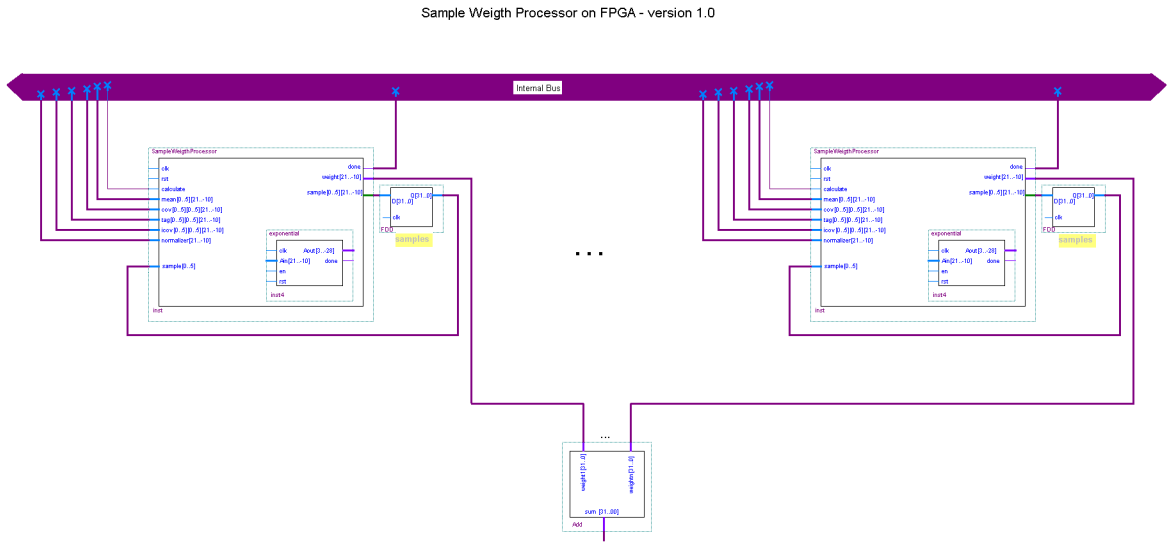


Figure 4.9: Integration of the sample weight processor (SWP) units into the particle filter logic (PFL). Single data, which is the same for all SWP operations, is transmitted via an internal bus. Multiple data like samples are received from a sample storage.

The sample weight processor (SWP) is a central component of the FPF. It contributes the most to the concurrent computation of the particle filter algorithm (PFA). While the place and function of the SWP in the architecture with respect to the PFA is straightforward,

the VHDL implementation of the SWP has to be done under a a lot of constraints. These constraints result in a violation of the mathematical theory on which the PFA is based on. The effects of these violations are analyzed and discussed later in this section. The SWP computes the line 4 of the PFA (see Algorithm Particle Filter (X_{t-1}, u_t, z_t)).

Algorithm Particle Filter (X_{t-1}, u_t, z_t):

1 $\overline{X_t} = X_t = \emptyset$
2 **for** $m = 1$ *to* M **do**
3 sample $x_t^{[m]} \sim p(x_t|u_t, x_{t-1}^{[m]})$
4 $w_t^{[m]} = p(z_t|x_t^{[m]})$
5 $\overline{X_t} = \overline{X_t} + (x_t^{[m]}, w_t^{[m]})$
6 **for** $m = 1$ *to* M **do**
7 draw i with probability $\propto w_t^{[m]}$
8 add $x_t^{[i]}$ to X_t
9 return X_t

Here the weight $w_t^{[m]}$ of a sample $x_t^{[m]}$ is given by the probability of the sensor measurement z_t, under the assumption that the vehicles state is given by $x_t^{[m]}$. For sensor observations a normal distributed error is assumed, where the bias is removed by software before transmission to the FPGA. Thus z_t is made up from a mean vector μ and the covariance matrix Σ. The probability $p(z_t|x_t^{[m]})$ is then given by the equation 4.3.

$$p(z_t|x_t^{[m]}) = det(2\pi\Sigma)^{-\frac{1}{2}} e^{\left(-\frac{1}{2}(x_t^{[m]}-\mu)\Sigma^{-1}(x_t^{[m]}-\mu)^T\right)} \tag{4.3}$$

The FPGA supports 6 dimensions but sensors do not necessarily provide information about all the dimensions. Therefore the FPGA must compute the weight with possibly incomplete information. This would require $2^6 - 1$ dedicated functions for every possible configuration of sensor inputs. In order to use just one function the following method has been developed that works as follows: The sensor data is transmitted as vector μ[5] of size m describing the mean of the measurement and a covariance matrix Σ[6] of size m, m describing the error of the measurement and an additional matrix T[7] of size m, m that indicates which dimensions are actually measured by the sensor. For example a SMR provides data on translation of the vehicle only which would result in:

$$z_t := \left[\mu : \begin{pmatrix} t_x \\ t_y \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \Sigma : \begin{pmatrix} \sigma_{t_x} & \sigma_{t_x,t_y} & 0 & 0 & 0 & 0 \\ \sigma_{t_y,t_x} & \sigma_{t_y} & 0 & 0 & 0 & 0 \\ 0 & & \frac{1}{2\pi} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2\pi} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2\pi} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2\pi} \end{pmatrix}\right], T := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

By multiplying the deviation between the mean μ and the system state x with the matrix T the same probability can be computed with one function instead of having to use

[5] μ contains 0 for unknown state variables.
[6] σ contains 1 for unknown state variables.
[7] The indexes for T are actually transmitted as a vector, but to save hardware resources the actual computation uses the matrix multiplication unit that is also used for the other vector multiplications with matrices.

individual functions for all possible combination of measured dimensions. The function is shown in Equation 4.4 and the proof is given later in this section.

$$p(z_t|x_t^{[m]}) = det(2\pi\Sigma)^{-\frac{1}{2}} e^{\left(-\frac{1}{2}((x_t^{[m]}-\mu)\cdot T)\Sigma^{-1}((x_t^{[m]}-\mu)\cdot T)^T\right)} \tag{4.4}$$

This equation is partially computed on the FPGA particle filter and partially on the software providing the sensor measurements.

1. $det(2\pi\Sigma)^{-\frac{1}{2}}$: The determinant of the covariance matrix Σ is calculated in software and transmitted to the FPGA. The reason is the computational complexity and the errors induced by the precision of the number representation.

2. Σ^{-1}: The inverse of the covariance matrix Σ is also computed in software and transmitted to the FPGA. The reason is the computational complexity and the errors induced by the precision of the number representation.

3. $(x_t^{[m]} - \mu) \cdot T$: The difference between the sensor observation mean μ and the sample $x_t^{[m]}$ followed by the multiplication with T is computed in the SWP.

4. e^y: The straightforward computation of exponential function e^y is computationally expensive and prone to rounding errors as the definition is:

$$e^y = \sum_{k=0}^{\infty} \frac{x^k}{k!}, y \in R \tag{4.5}$$

   Therefore a mathematical trick is used. The exponential function is computed on the SWP by means of a precompiled look-up table that contains 360 values in the range of $[e^0, e^{ln(2)}]$, in steps of 0.002. This is possible because the argument y for the exponential function is always negative which results in a in $e^y \in [0, 1]$. The range of the output function and the reasonable range of the input values allow a mathematical conversion to approximate the value e^y with a minor loss of precision if we assume that y is the sum of x and z:

$$e^y = e^{x+z} = e^x \cdot e^z \tag{4.6}$$

   where we can, because of

$$e^y = e^{y-p\cdot ln(2)} \cdot 2^p \tag{4.7}$$

   , approximate e^z by 2^p

$$e^y = e^{x+z} = e^x \cdot 2^p \tag{4.8}$$

   given that $z = p \cdot ln(2)$ and $x = y - z; x \in [0, ln(2)]$, so p most be chosen such that the condition $x \in [0, ln(2)]$ is satisfied.

   If $y \in [0, ln(2)]$ than p is 0 and $e^y = e^x * 2^0$, but this is always true because y is always negative. Because y is negative p must also be negative to make x positive.

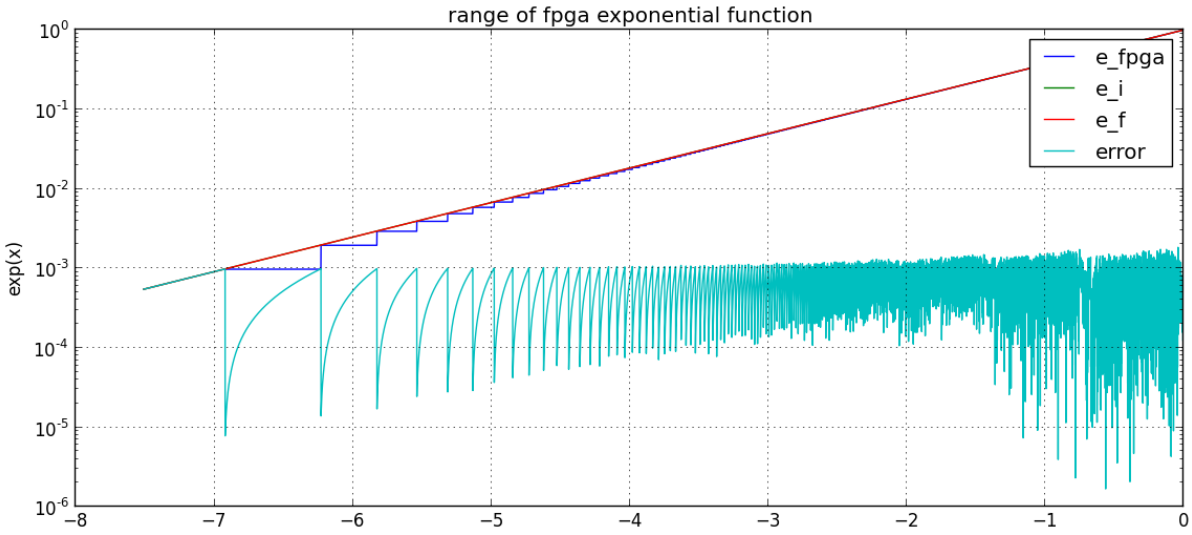


Figure 4.10: Comparison of the FPGA exponential approximation to standard floating point precision. The turquoise line marks the difference between the LUT and the exp function. The upper bound for the error is about 0.001, but the approximating function decreases monotonically.

If $y \in [-0, -ln(2)]$ than p is -1 and $x = y + ln(2)$ and $e^y = e^x * 2^{-1}$.

If $y \in [(p-1) \cdot ln(2), p \cdot ln(2)]$ than $x = y - p * ln(2), x \in [0, ln(2)]$ and $e^y = e^x * 2^p$.

There is no default value $p_{default}$ for p that satisfies the condition $x = y - p \cdot ln(2), x \in [0, ln(2)]$ in all cases, unless a sensor data normalization is performed to force y to remain in the interval $[(p_{default} - 1) \cdot ln(2), p_{default} \cdot ln(2)]$. This could be done by dividing y by a reasonable factor precalculated in software from the sensor observations uncertainty that has to be transmitted with each sensor observation. However this can be hardly done automatically for new sensors. In favor to maintain the generic processing capabilities of the FPGA particle filter p is determined by an iterative algorithm that assigns p in such a way that the requirement $x \in [0, ln(2)]$ is satisfied. The algorithmic implementation uses a precompiled values with a range of $[-ln(2), 30 \cdot ln(2)]$.

Figure 4.10 shows that the error of the approximating function decreases proportionally to the standard exp function. More important the fixed point approximation of the exponential function decreases monotonically and does not exceed the floating point approximation of the exponential function.

SWP error analysis

The error induced by fixed point usage has been analyzed using a simulation of the sample weight processor. The simulation results show that this error varies for different sensor types. The marginalized error distributions for surface movement radar (SMR), laser scanner and universal medium range radar (UMRR) are shown in Figure 4.11. Noteworthy is the residual 100% error for some sensors. There are a number of causes contributing to the error.

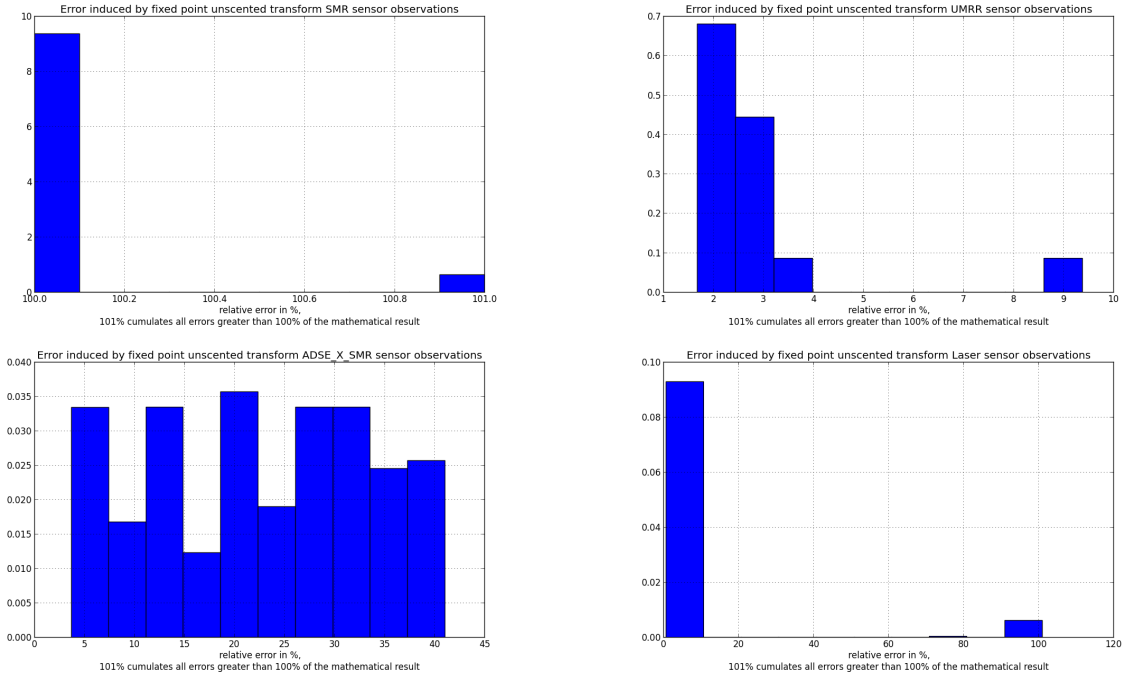


Figure 4.11: Error distribution of the fpga sample weight computation of observations mapped to the cartesian coordinate system.

1. FPGA exponential calculation: The input value to the exponential function may be rounded in such a way that a part of the fraction is cut away. As the input value is always negative this causes the result to be different from lower values the flat parts of the output function shown in Figure 4.10 appear as plateaus in the probability density function as shown in Figure 4.12

2. Inverse covariance matrix: Values may be rounded down to 0, or overflow in the negative range or both. Depending on the values in the covariance matrix the inverse covariance matrix may contain values too big or too small for the fixed point representation. This results in a wrong input value for the exponential calculation.

3. Normalizer: The normalizers value may be cut away to 0, or overflow in the negative range or both.

Figure 4.12 shows the difference between the 64bit floating point calculation and the fixed point calculation. The effects of these errors are :

1. Zero Weights: The weights of the samples may be assigned to a weight of 0. This will affect the resampling procedure in many ways. In the worst case the procedure will not be terminating. The solution of this problem is discussed later in this section.

2. Plateaus in the PDF: Because the gradient of the PDF might be encoded in the fractional part of the fixed point number the bits that hold the information might be cut away, which results in a more or less coarse stair function. The effect on the tracking accuracy is not as big as one might expect because the mean value of the extracted probability density is not significantly influenced.

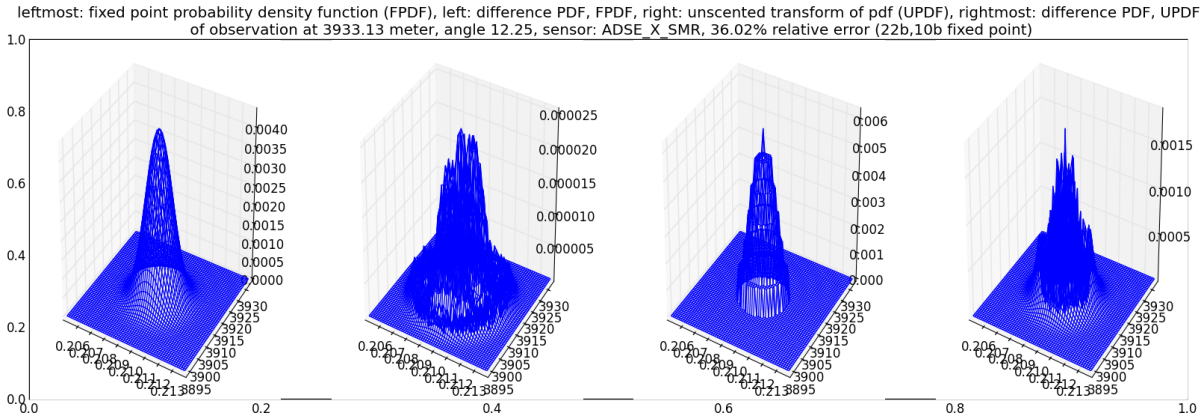


Figure 4.12: Error induced by the fixed point computation of polar probability density functions mapped to cartesian coordinates. The image shows the detailed fixed point polar and Cartesian probability density as well as the error compared to the 64bit floating point probability density. For this plot the probability density function has been sampled with 900 data points in an isometric pattern over the interval $[\mu - 5\sigma, \mu + 5\sigma]$.

3. Unreasonable probabilities: In some rare cases the PDF is complete wrong. The reasons lie in the insufficient dynamic range and precision of the fixed point number representation that cannot cope with extreme values in the covariance matrix. Although this is a systematic error, the noise on the observations usually prevents a series of this kind of error and the convergence of the CPFA is preserved.

A resolution that addresses all of these issues is the usage of an alternative fixed point representation. A different uniform representation used for all calculations is the quick solution. The other solution is the time consuming implementation of fixed point representations customized to the needs of each step of the computation. Figure 4.13 shows the error depending on the distance for different sensors and different uniform fixed point representations. Apparently not every fixed point pattern is equally suited for each sensor, depending on the characteristics of the individual sensors. Nevertheless the results suggest that it is possible to work with a 16.6 bit fixed point pattern for both laser scanner and UMRR sensor, which requires significantly less resources than the 22.10 bit implementation used for the prototype.

Treatment of Zero Weights

The treatment of zero weights is, for most application scenarios, also possible by the addition of a negligible likelihood, which can be realized with no computational extra effort by setting the least significant bit per default. The addition of a negligible likelihood p_n to the weight w_t of the samples x_t does not only remove the arithmetic errors but affects the algorithm in multiple ways.

Impact on the Importance Sampling: Importance sampling means to compute an expectation over a PDF f (given by the sensor observation) from a sample set with a density of another PDF g. One is interested in the expectation that $x \in A$, where A is

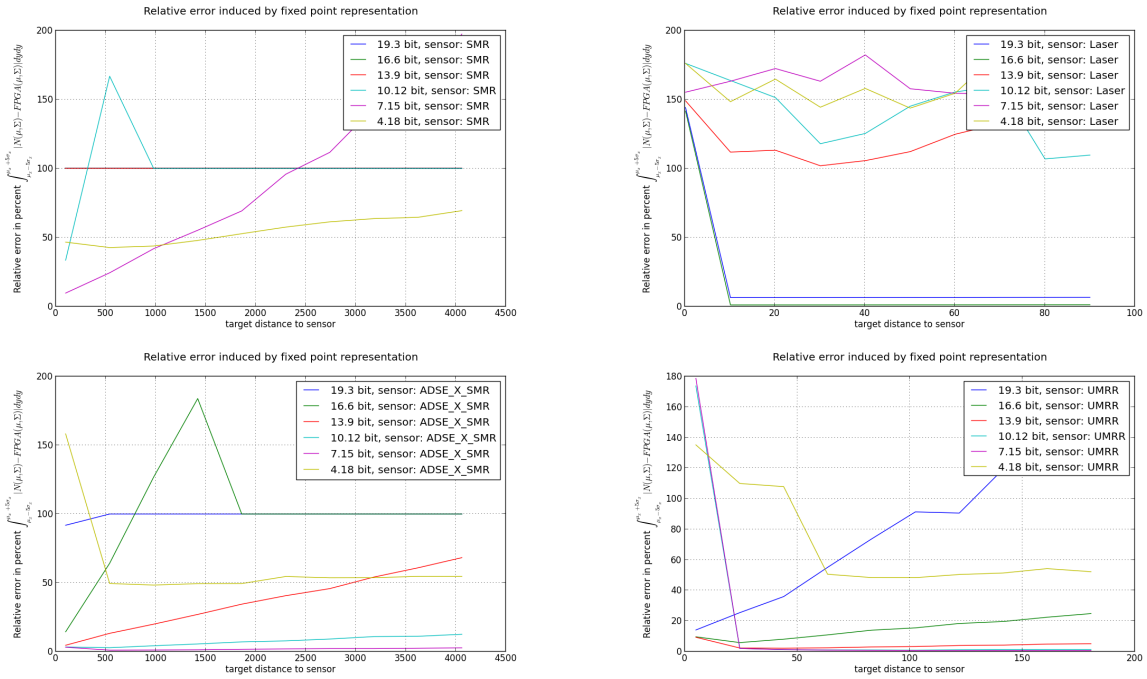


Figure 4.13: Influence of the integer and fractional part bit length of the fixed point number representation on the marginalized error induced by the fixed point computation of polar PDFs mapped to Cartesian coordinates.

an interval $A \subset dom(X)$. Using an indicator function I that is 1 if its argument is true and 0 otherwise, one can express this probability as an expectation over g.

$$E_f[I(x \in A)] = \sum_{x \in X} f(x) I(x \in A) \tag{4.9}$$

$$= \sum_{x \in X} \frac{f(x)}{g(x)} g(x) I(x \in A) \tag{4.10}$$

$$= \sum_{x \in X} w(x) g(x) I(x \in A) \tag{4.11}$$

$$= E_g[w(x) I(x \in A)] \tag{4.12}$$

Here $w(x) = \frac{f(x)}{g(x)}$ is a weighting factor that accounts for the mismatch between f and g. For this equation to be correct we need $f(x) > 0 \rightarrow g(x) > 0$. The condition $g(x) > 0$ is given because $f(x)$ is only calculated for samples which implies $g(x) > 0$. As explained earlier $f(x) > 0$ is not always true in the FPGA implementation of the particle filter, but a work around is used to ensure that there are always some samples where $f(x) > 0$ is true. It can be shown that the normalized weight mass of the sample set converge to the integral of f under A.

$$[\sum_{m=1}^{M} w^{[m]}]^{-1} \sum_{m=1}^{M} I(x^{[m]} \in A) w^{[m]} \rightarrow \int_A f(x) dx \tag{4.13}$$

After adding p_n the approximated PDF is not $E_f[I(x \in A)]$ anymore, but a compound function over the expectations of f and g, where the g is scaled by the negligible likelihood p_n.

$$
\begin{aligned}
& E_g[(w(x)+p_n)I(x\in A)] && (4.14)\\
= & \sum_{x\in X}(w(x)+p_n)g(x)I(x\in A) && (4.15)\\
= & \sum_{x\in X}(\frac{f(x)}{g(x)}+p_n)g(x)I(x\in A) && (4.16)\\
= & \sum_{x\in X}(f(x)+p_n g(x))I(x\in A) && (4.17)\\
= & \sum_{x\in X}\Big(f(x)I(x\in A)+p_n g(x)I(x\in A)\Big) && (4.18)\\
= & \sum_{x\in X}f(x)I(x\in A)+\sum_{x\in X}p_n g(x)I(x\in A) && (4.19)\\
= & \sum_{x\in X}f(x)I(x\in A)+p_n\sum_{x\in X}g(x)I(x\in A) && (4.20)\\
= & E_f[I(x\in A)]+p_n E_g[I(x\in A)] && (4.21)
\end{aligned}
$$

The PDF approximate by the weight mass of the samples in the sample set is now tainted with the proposal distribution.

$$
\begin{aligned}
& [\sum_{m=1}^{M} w^{[m]}+p_n]^{-1}\sum_{m=1}^{M} I(x^{[m]}\in A)(w^{[m]}+p_n) && (4.22)\\
& \qquad\qquad \rightarrow \int_A f(x)dx+p_n\int_A g(x)dx && (4.23)
\end{aligned}
$$

As a consequence some of the samples in the proposal distribution $\overline{X_t}$ that would not be included into the sample set X_t by the original algorithm are now included in X_t. Also during the normalization of the sample weights p_n sums up for each sample resulting in a flatter PDF then during the normalization without the usage of p_n. These two effects result in a slower rate of convergence of the estimated object position to the true object position. This is not a problem for stationary objects or slow moving objects, but it can be a problem if the ratio between object dynamics and sensor update rate is too big. The actual error induced by p_n is relative to the quality of the proposal distribution, the better the quality of the proposal distribution the less the error induced by p_n. To counter such effects the FPGA implementation replaces a small fraction of the compound distribution with samples generated from measurements after the importance sampling increasing the density of the function according to f at random points.

Impact on the Data Association: As the sample weights are also used to determine the independent data association probability, the error induced by the addition of the negligible likelihood needs to be taken into account during the calculation of the joint data association probability, which is needed to assign detected features to tracks.

Uniform PDF Calculation

As mentioned earlier in this section, the efficient realization of the computation of the sample weights, given sensor observations of arbitrary subsets of the samples state space,

is only possible by an extension of the standard way to calculate normal probability density functions. This extension requires the sensor observation to include a Tag matrix T that can be transmitted as a vector due to the nature of the matrix that allows the reconstruction from the vector. In the following text the proof of the underlying mathematical model is given.

Proposition: Consider M to be an n-dimensional frame referred to as measuring space, which is a subspace of an m-dimensional frame Z referred to as state space. Further consider f to be a function that maps the index $i \in \{1, \ldots, n\}$ of every dimension in M to the index $j \in \{1 \ldots n\}$ of the corresponding dimension in Z, and $s_o = (v, A)$ to be a Normal distribution with the mean $v \in M$ and the n, n-covariance matrix A, which is a sample point in M. Furthermore consider $z = (\mu, \Sigma)$ to be a normal distribution with a mean $\mu \in Z$ and a m, m-covariance matrix Σ, defined as an identity matrix divided by 2π that fulfills $\Sigma_{f(i),f(j)} = A_{i,j}$. Now consider T to be a m, m-matrix where all elements $T_{f(i),f(i)}$ have the value 1, and other elements have the value 0. Consider x to be a sample in Z that satisfies the condition $x_{f(i)} = s_i$ and $x_{\notin f(i)} = 0$. Than the following equation is valid:

$$det(2\pi A)^{-\frac{1}{2}} e^{-\frac{1}{2}(s-v)A^{-1}(s-v)^T} = det(2\pi\Sigma)^{-\frac{1}{2}} e^{-\frac{1}{2}((x-\mu)T)\Sigma^{-1}((x-\mu)T)^T} \tag{4.24}$$

Proof: The proof is divided in two parts, one for the front part of the equation and one for the back part of the equation.

Proof - Part 1: In this part the following equation is shown to be valid.

$$det(2\pi A)^{-\frac{1}{2}} = det(2\pi\Sigma)^{-\frac{1}{2}} \tag{4.25}$$

The proof is based on the following well known statements that can be found in mathematical textbooks, e.g. [6]

1. The determinant of a triangular matrix is the product of the diagonal elements $det(M) = m_{11} \cdot m_{22} \cdot \ldots \cdot m_{nn}$.

2. Every square matrix can be transformed to a triangular matrix using elementary row transformations, where the multiple of a row is added to another row.

3. The determinant is invariant with respect to the elementary row transformations from the previous statement. If a square matrix M is transformed into another square matrix M' than $det(M) = det(M')$ is valid.

The diagonal elements in Σ that are not connected to elements in A are 1 after the multiplication with 2π as in Equation 4.25, all other elements in Σ that are not connected to elements in A are 0. Therefore these elements do not infer during the computation of the determinant. The determinant is therefore set only by the diagonal elements in A and the Equation 4.25 is valid. In conclusion the scaling factor in Equation 4.24, that sets the maximum value at $x - \mu = 0$, is the same for booth sides of the transformations.

Proof - Part 2a: The first thing to show is that the elements that are mapped by f from A to Σ have the same values on booth side of the equation after the inversion of the covariance matrices, so:

$$a_{ij}^{inv} = \sigma_{f(i)f(j)}^{inv} \tag{4.26}$$

The rows and columns in Σ do only hold elements from A else the elements are 0 or, on the diagonal, $\frac{1}{2\pi}$. Thus the rows and columns that hold values mapped from A are linear independent from rows and columns that do not hold elements mapped from A.
During the calculation of the inverse matrix with the Gauss-Jordan-Algorithm linear independent vectors do not interfere with each other. Therefore the elements is the inverse covariance matrix Σ^{-1} are only the result of calculations involving the elements in Σ that have been mapped from A. What remains to be shown is that the rest of the elements in the inverse covariance matrix does not affect the argument of the exponential term in Equation 4.24.

Proof - Part 2b: Here the following equation must be shown to be valid, based on Part 2a of the proof.

$$(s-v)A^{-1}(s-v)^T = ((x-\mu)T)\Sigma^{-1}((x-\mu)T)^T \tag{4.27}$$

For this part of the proof the definition of the elements in x, μ, Σ, and T by the function f is crucial. Therefore the definition is formally reviewed here.

$$x = (x_{11} \dots x_{1m}) \tag{4.28}$$

$$v = (v_{11} \dots v_{1n}) \tag{4.29}$$

$$s = (s_{11} \dots s_{1n}); s_{1i} = x_{1f_{(i)}} \tag{4.30}$$

$$\mu = (\mu_{11} \dots \mu_{1m}); \mu_{1i} = \begin{cases} 0 & i \notin f[I] \\ v_{1f_{(i)}^{-1}} & i \in f[I] \end{cases} \tag{4.31}$$

$$\sigma_{ij}^{-1} \in \Sigma = \begin{cases} \frac{1}{2\pi} e_{ij} & i,j \notin f[I] \\ a^{-1}_{f_i^{-1} f_j^{-1}} & i,j \in f[I] \end{cases} \tag{4.32}$$

$$t_{ij} \in T = \begin{cases} 0 & i,j \notin f[I] \\ 1 & i,j \in f[I] \end{cases} \tag{4.33}$$

For the proof the intermediate results r, y, l, that follow from the above definitions, are needed.

$$r = (r_{11} \dots r_{1n}); r_{1i} = s_{1i} - v_{1i} \tag{4.34}$$

$$y = (y_{11} \dots y_{1m}); y_{1i} = x_{1i} - \mu_{1i} \tag{4.35}$$

$$l = (l_{11} \dots l_{1m}); l = yT; l_{1j} = \sum_{i=1}^{m} y_{1i} t_{ij} \tag{4.36}$$

Because of the definition of y and μ the following equation is valid.

$$y_{1i} = \begin{cases} x_{1i} - \mu_{1i} & i \notin f[I] \\ s_{1f_i^{-1}} - v_{1f_i^{-1}} = r_{1f_i^{-1}} & i \in f[I] \end{cases} \tag{4.37}$$

Using the definition of T, l is:

$$l_{1i} = \begin{cases} 0; i \notin f[I] \Rightarrow t_{ij} = 0 \\ \sum_{i=1}^{m} r_{1i} = r_{1i}; i \in f[I] \end{cases} \tag{4.38}$$

Which allows to rewrite Equation 4.27 as:

$$rA^{-1}r^T = l\Sigma^{-1}l^T \tag{4.39}$$

For the further proof vectors g and h are needed:

$$g = (g_{11} \dots g_{1n}); g = rA^{-1}; g_{1j} = \sum_{i=1}^{n} r_{1i}a_{ij}^{-1} \tag{4.40}$$

$$h = (h_{11} \dots h_{1m}); h = l\Sigma^{-1}; h_{1j} = \sum_{i=1}^{m} l_{1i}\sigma_{ij}^{-1} \tag{4.41}$$

which result, with respect to the definition of l and Σ in

$$h_{1j} = \begin{cases} \sum_{i=1}^{m} r_{1f_{(i)}^{-1}} a^{-1}_{f_{(i)}^{-1} f_{(j)}^{-1}} = g_{1f_{(j)}^{-1}} & j, i \in f[I] \\ 0 & j \notin f[I] \end{cases} \tag{4.42}$$

$$\tag{4.43}$$

Therefore Equation 4.27 is

$$gr^T = hl^T \sum_{i=1}^{n} g_{1i}r_{i1}^T = \sum_{i=1}^{m} h_{1i}l_{i1}^T \tag{4.44}$$

$$\tag{4.45}$$

and with respect to Equation 4.42 it follows that:

$$\sum_{i=1}^{m} h_{1i}l_{i1}^T = \begin{cases} \sum_{i=1}^{m} g_{1f_{(i)}^{-1}} r^T_{1f_{(i)}^{-1}} & i \in f[I] \\ 0; i \notin f[I] \end{cases} \tag{4.46}$$

The inverse function f^1 is surjective with respect to the image set $f^{-1}[f[I]]$ and it follows that:

$$((x-\mu)T)\Sigma^{-1}((x-\mu)T)^T = \sum_{i=1}^{m} h_{1i}l_{i1}^T = \sum_{i=1}^{n} g_{1i}r_{i1}^T = (s-v)A^{-1}(s-v)^T \tag{4.47}$$

Because the result of the terms considered in both parts of the proof are equal the results of both sides of the Equation 4.24 are equal.

4.2.5 DSP-Usage

Many FPGAs feature embedded digital signal processing resources like multipliers that are capable of implementing the simple multiplication operation commonly used in typical DSP functions. Cyclone III devices offer up to 288 embedded multiplier blocks and support the following modes: one individual $18-bit \times 18-bit$ multiplier per block, or two individual $9-bit \times 9-bit$ multipliers per block, see Figure 4.14. Multipliers can be cascaded to support wider bit widths. Multipliers can also be inferred directly from VHDL or Verilog source code. The prototypical implementation uses the multipliers implicit, as the compiler integrates them to save LEs and for a faster computation of the results.

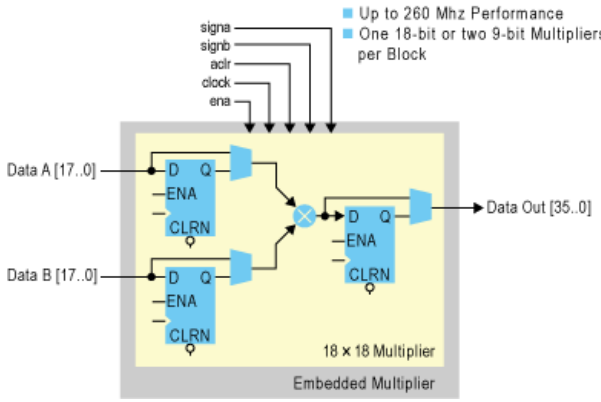


Figure 4.14: DSP Resource [14]

4.2.6 NIOS Processor

The NIOS processor is a softcore processor provided by Altera as a compilable IP. The NIOS can be configured with the qsys system builder to provide support for extensive debugging and additional features.
In the context of this work the most basic variant of the debugging support for the NIOS has been used, providing only a JTAG target connection, download of software, and the support of software breakpoints. Although the PFL works with the most basic variant of the NIOS, in order to support a broad range of peripheral interfaces the most complex NIOS variant is used, the NIOS II/f. This variant features a single core 32-bit RISC processor with instruction cache, data cache, dynamic branch prediction hardware multiply/divide, and barrel shifter.

4.2.7 Memory Controller

The memory controller implements the protocol and bus used by the NIOS processor and the particle filter logic to exchange data. On side of the NIOS the data of the bus is readable at a specific memory address that can be accessed conveniently using C-Macros generated by the Altera System Builder as shown in Listing 4.1.
Because the particle filter logic is much faster then the software execution on the nios, a handshake protocol, see Figure 4.15, had to be implemented to ensure the correct transmission of the data between the nios and the particle filter logic. The handshake

Listing 4.1: NIOS c-code to write data. This code-snippet is part of the main routine executed by the NIOS.

```
//send data to buffer_out (to the logic)
IOWR_ALTERA_AVALON_PIO_DATA(DATA_BASE,n);
//data_enable =1 to indicate to the logic that a new value is available
IOWR_ALTERA_AVALON_PIO_DATA(DATA_AVAILABLE_BASE,1);

while(!end) { //it remains some data to send
    while (!IORD_ALTERA_AVALON_PIO_DATA(DATA_READ_BASE)) ;//wait until data has been read by the FPGA
        //usleep(2000000); //wait 0.1s

    //data sent by nios has been read by FPGA
    IOWR_ALTERA_AVALON_PIO_DATA(DATA_AVAILABLE_BASE,0);
    n = strtol(p+1, &p, 10);  //10 = base
    //no more value available:
    if ('\n' == *p) {
        end=1;
    }
    //send data to buffer_out
    IOWR_ALTERA_AVALON_PIO_DATA(DATA_BASE,n);
    IOWR_ALTERA_AVALON_PIO_DATA(DATA_AVAILABLE_BASE,1);
}
```

protocol works in the following way. When the PFL want's to transmit data to the NIOS it sets a data available signal and puts the data on the bus. The NIOS reads the data and acknowledges that it has read the data by raising the data read signal. The PFL reacts by lowering the data available signal, which in turn causes the NIOS to lower the data read signal. If the NIOS sends data to the PFL the protocol works the same way.

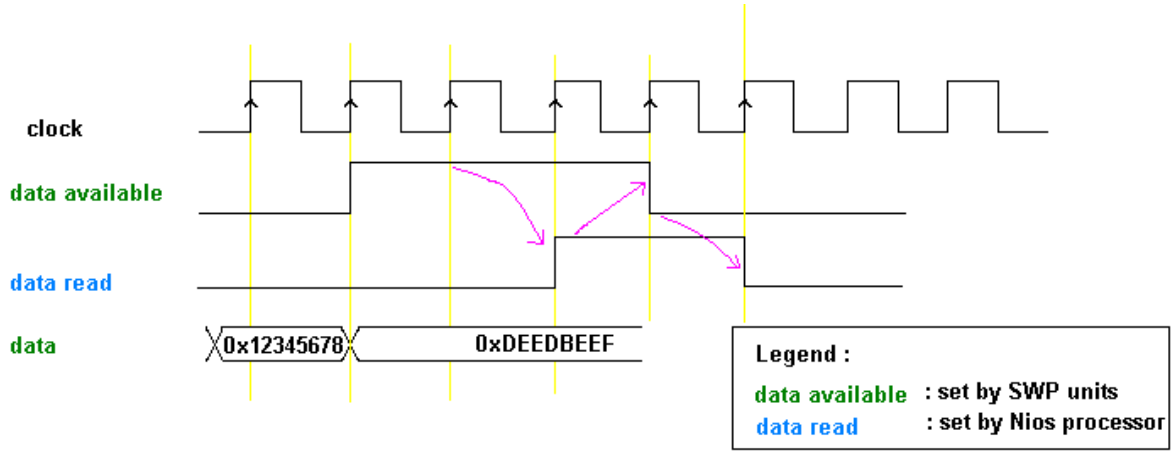


Figure 4.15: Memory controller bus protocol except [14]

4.2.8 Input Memory

The input memory is needed to store the sensor data transmitted to the FPGA in such a way that the data is easily accessible from the particle filter logic. Values stored here are:

- Tag Matrix: Unless the Tag matrices for every possible measured subset of the estimated state space are stored in hardware, or implement the complex and resource consuming generation algorithm in hardware, a software realization must be used to compute the matrix. For these reasons the matrix is communicated with the sensor observations messages.

- Determinant: The calculation of the determinant must be only done once and is not dependent on the track. Also the calculation requires a rather complex algorithm. Therefore it is computed by software and transmitted with the sensor observation in order to avoid unnecessary redundant computations and save power and hardware resources.

- Inverse Covariance Matrix: As for the determinant, the computation of the inverse matrix is complicated and would require a lot of hardware resources. Also the Matrix is independent of the tracks. It is, like the determinant, transmitted with the sensor observation in order to avoid unnecessary redundant computations and save power and hardware resources.

4.2.9 Sample Storage

Unless all samples are stored in the PUs, as in the case of full parallelization, a the storage for the samples is needed. Each sample $x_t^{[m]}$ in the sample set X_t requires $7 \cdot 32 = 224$ bit for 6 state space variables and the sample weight $w_t^{[m]}$. The table shows the storage capacity required for different sample set sizes. The LE usage estimation and memory cell usage estimation are based on the compilation results of the prototype on an Altera Cyclone III FPGA. Because the samples are stored in the M9K memory units there is no direct LE usage by the samples. Additional LE usage not shown in the table results from the logic needed to load and store samples. Embedded memory structures as featured by the Cyclone III device family are a common concept to satisfy the memory needs of embedded designs in FPGAs, without sacrificing the more versatile LE resources for such a single use task.

Type of Sample	Number of samples	Sample set size	LE usage estimation	Memory Cell Usage Estimation
uniform	128	28627 bit	0 (38k)	4 M9K
uniform	256	57344 bit	0 (76k)	7 M9K
uniform	512	114688 bit	0 (153k)	14 M9K
uniform	1024	229376 bit	0 (307k)	28 M9K
uniform	2048	458752 bit	0 (614k)	56 M9K
uniform	4096	917504 bit	0 (1229k)	112 M9K
RoboCup	128	11008 bit	0 (14k)	2 M9K
RoboCup	256	22016 bit	0 (28k)	3 M9K
RoboCup	512	44032 bit	0 (56k)	6 M9K
RoboCup	1024	88064 bit	0 (112k)	11 M9K
RoboCup	2048	176128 bit	0 (224k)	22 M9K
RoboCup	4096	352256 bit	0 (448k)	43 M9K

Table 4.2: Memory usage for different sample set sizes. The LE usage estimation value in brackets shows the estimated LE usage if no internal memory blocks are available.

An alternative solution would distribute the samples to an input cache and output cache and an RAM external to the FPGA. During the partially serial calculation of the sample weights the input cache will be refreshed from the RAM and the output cache will be written to the RAM. This solution requires only a fixed amount of memory cells or LEs at the cost of addition resources used for the RAM interface and the additional control logic to ensure the memory consistency. The primary advantage of this solution is the ability to implement particle filters with sample set sizes that require more memory than

available as memory cells. Additional benefits result in the ability to concurrently access the sample set by software, for alternative computations like visualization or analysis.

4.2.10 Control by Finite State Machine

The top level control is done by a finite state machine (FSM), that controls the interaction of the processes of the components, based on its own state, the input given from the outside and the output of the components. Thus it realizes the execution of the Algorithm Concurrent Particle Filter (X_{t-1}, u_t, z_t) and governs the execution of the algorithms by the subordinate components that perform the low-level computations of the particle filter.

4.2.11 Resampling

The resampling or importance sampling is the part of the algorithm that allows the usage of a limited number of samples to estimate the probability density in arbitrary large spaces. The algorithm condenses the samples in the sample set to the, given the available background and sensor information, regions of the system state space with the highest probability density. These are encoded in lines 6-8 in the particle filter algorithm:

Algorithm Particle Filter (X_t, W_t)**:**

```
1 X̄_t = X_t = ∅
2 for m = 1 to M do
3 |   sample x_t^[m] ~ p(x_t | u_t, x_{t-1}^[m])
4 |   w_t^[m] = p(z_t | x_t^[m])
5 |   X̄_t = X̄_t + (x_t^[m], w_t^[m])
6 for m = 1 to M do
7 |   draw i with probability ∝ w_t^[i]
8 |   add x_t^[i] to X_t;
9 return X_t
```

For implementations under real time constraints one has to consider that the straightforward implementation of line 7 requires to loop for possibly a very long time over the sample set and compare the weights with random numbers. This drawback of the algorithm led to a number of real time capable alternative resampling algorithms. From these the low-variance sampler algorithm had been selected for the implementation of the resampling. The low-variance sampler has three advantages. First, the complexity of the low-variance sampler is of $O(M)$ for M samples. Second the space of samples is covered more systematically than by the random sampler. Third if all samples have equal weights, all samples are kept.

Algorithm Low Variance Sampler (X_{t-1}, u_t, z_t)**:**

$\overline{X_t} = \emptyset$
r = random(0;M-1)
c = $w_t^{[1]}$
i = 1
for $m = 1$ *to* M **do**
  $U = r + (m-1) \cdot M^{-1}$
  while $U > c$ **do**
    i = i + 1
    c = c + $w_t^{[i]}$
  add $x_t^{[i]}$ to $\overline{X_t}$
return $\overline{X_t}$

The algorithm requires random numbers that are generated by a linear feedback shift register (LFSR). The LFSR is fed by its own state, driven by the exclusive-or (xor) of some bits of the overall shift register value. Figure 4.16 illustrates the function of the register.
Determined by the deterministic working of the LFSR the stream of random numbers produced is deterministic and the same for the same initial value. Therefore the register will produce a cyclic output sooner or later. However a carefully designed feedback function can produce a very long sequence of values that appear to be random. The LFSR is used during the resampling step as well as during the generation of samples from measurements.

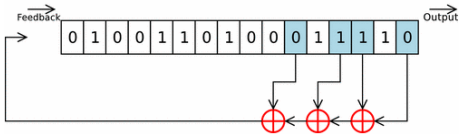


Figure 4.16: A 16 bit linear feed back shift register. The register is used to generate long sequences of pseudo random numbers [14].

4.3 Performance

This section discusses the performance of the PFA with respect to the speed, accuracy, and power consumption of the prototypical implementation. The analysis of the performance is done with the goal to make a cautious assumption about the benefits a specialized tracking hardware, similar to a GPU or a mathematical coprocessor, might offer to embedded systems that depend on tracking to monitor their environment. Therefore the comparison is done based on the implementation of a general purpose processor (GPP), the NIOS II/e, and an implementation of the concurrent particle filter algorithm (CPFA) on the Cyclone III FPGA. The GPP executes a software implementation of the particle filter algorithm that is compared with the execution of the hardware implementation of the

CPFA in terms of speed, resource requirements, power consumption, and heat dissipation of the system. To compare the performance with that of a highly optimized GPP the software implementation has been tested on a laptop with an intel I7 3537U processor running on battery at 750MHz.

4.3.1 System Setup

The system was tested using data recorded in a RoboCup environment and simulated sensor data from an airport environment. The RoboCup environment data where positions of moving robots detected by a ceiling camera above a RoboCup soccer field. The airport environment data where simulated Surface Movement Radar (SMR) measurements and Universal Medium Range Radar (UMMR) measurements of moving vehicles. The dynamic scaling was not implemented for the test but the addition of the negligible likelihood was.
The particle filters used for the analysis has 128 samples. Each sample has 6 variables representing the state of the sample and 1 variable for the weight of the sample. Although the software implementation is roughly equivalent of the hardware implementation there are some differences between the two implementations which affect the order of execution and the representation of numbers.
The software implementation of the particle filter algorithm is a serial implementation of the basic particle filter algorithm with a low variance resampling. The essential dimensions of data types are the same as in the hardware implementation of the CPFA. Relevant numbers for the important calculations are represented by the float data type, which is - from the author's experience - the most common approach for software implementations of particle filters. The software was compiled in release mode with the standard optimization options from the NIOS II software tools and Visual Studio 2013.
The NIOS II/e is the most basic variant of the NIOS II processor. It is a 32bit RISC processor that has no dedicated hardware for mathematical calculations, no branch prediction, no instruction cache, and no data cache. Because the NIOS II/e is a very simple processor, the software implementation has also been tested on a Lenovo Yoga 13 laptop, on one core of the Intel I7 3537U processor running on battery power at 750 MHz.
The CPFA implementation uses 4 units to generate samples and compute sample updates (see 4.2.1 particles_inst) , 1 unit to compute the sample weights (see 4.2.1. SWP_inst), and 1 unit to do the importance sampling (see 4.2.1 resample). The algorithm is executed in a partial concurrent pipeline as shown in Figure 4.17. The importance sampling on the hardware is done only if needed, and is executed concurrently to the update and calculation of the sample weights. Using 2 alternating sample sets, a sample is selected just in time from the old set, updated, weighted and stored to the new set. In the next observation update the new sample set becomes the old sample set.

4.3.2 Data Acquisition

Speed

The speed of the most time consuming operations, the initialization of the sample set from an observation, the calculation of the sample movement update, calculation of the

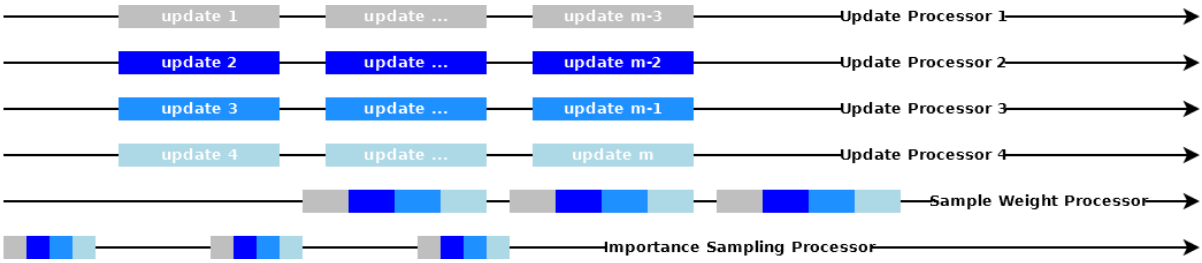


Figure 4.17: Timing of the pipelined concurrent execution with interleaved resampling of the PFA as implemented by the CPFA implementation.

sample weights, and the importance sampling have been measured using the Quartus-Signaltap tool. The Signaltap allows to record the state of selected registers and pins, after a previously set trigger condition is met, in real time for a limited amount of time or number of signal changes. In order to measure the time needed reliable and accurate, additional clock signals, with clock cycles of 0.1 MHz and 0.01 MHz, had to be defined and linked to dedicated output pins. The main clock was 50 MHz during the data acquisition. The time needed for the execution of certain functions by the software was measured by setting the state of dedicated output pins that have been added to the NIOSII VHDL entity for the communication with external hardware or other VHDL entities. The according pin was set to 1 as the first instruction of the respective function body and set to 0 before return instructions of the respective function body. Either edge in the pins signal was set as a trigger condition and the time needed by the functions was then calculated from the above mentioned dedicated extra clock signals and signal of the pin used to signal the functions state.
On the laptop functions from the chrono package of the c++ stl were used to estimate the average execution time from thousands of tracker updates.
The VHDL entities of the CPFA implementation has dedicated control signals to coordinate the components that use a strobe-protocol. The time needed for the operations mentioned above was measured using these signals as triggers and monitoring them, and also monitoring the dedicated extra clock signals.

Accuracy

To determine the accuracy of the CPFA implementation and the software solutions the output has been compared against a MatLab simulink implementation of the algorithm.

Resource Usage

The source of the data about the used resources is the Quartus software that provides detailed reports about the resources used by each VHDP entity and each IP. Information on the I7 3757 U and similar devices are provided by Intel [51].

3.3 Energy Consumption and Heat Dissipation

Accurate values for energy consumption and heat dissipation could not be obtained by measurement during the development of the CPFA. Therefore the discussion of these

parameters is based on simulation results, and the clock frequency and resources needed by the different solutions to compute the particle filter algorithm at the same speed, which is estimated from the data processing speed and resource usage. On the laptop the power consumption was estimated using the Joulemeter software from Microsoft Research.

4.3.3 Prototype Test Results

Speed

Timing simulation result of the particle filter logic indicated that the particle filter could integrate features at a rate of 5MHz with maximal parallelization. Since this update rate is not needed for most applications, the computational architecture has been adapted to support partial parallelization of the particle filter algorithm with 4 to 1024 samples. Still the analysis of the time needed for the operations show that there are significant differences in the time needed to compute the various parts of the algorithm between the NIOS II/e and the CPFA implementation. Depending on the operation the speed up factor may range from 196 to 3600. The most relevant factor is the speed up during the update of the tracker which is the most common operation. Here the CPFA is app. 3000 times faster.The Intel I7 3537 is about 2.6 times faster than the CPFA implementation when performing the update of the tracker. The measurements do comply with by the specifications of Intel and Altera that estimate the NIOSII/s at about 7-8 MIPS at 50 MHz and the I7 series at about 200k MIPS at 3 GHz [51].

Operation / System	NIOS II/e	CPFA	SU CPFA	I7	SU I7
Init tracker	0834.0 ms	0.3000 ms	2780	0.11 ms	2.7
Update tracker	1340.0 ms	0.4500 ms	2978	0.175 ms	2.56
Init single sample	0008.0 ms	0.0030 ms	2666	-	-
Weight single sample	0009.0 ms	0.0025 ms	3600	-	-
Update single sample	0000.5 ms	0.0005 ms	1000	-	-
Upd. + weight single sample	0009.5 ms	0.0030 ms	3166	-	-
Importance sampling	0074.7 ms	0.4400 ms	169	0.084 ms	52.6

Table 4.3: Execution speed of hardware and software tracker implementations. SU CPFA denotes the speed up factor from NIOSII/e to the CPFA i implementation, and SU I7 denotes the speed up factor from the CPFA implementation to the particle filter software executed on the I7.

Not included are the times needed for the communication with superordinate instances, which strongly depend on the communication protocol and other factors that vary for the different application scenarios. To estimate the performance when tracking multiple objects software simulations of the FPGA implementation had to be used. The FPGAs where simulated on different PCs and connected via Ethernet. It was found that the speed did not significantly decrease when tracking up to 4 targets. A test for more targets was not possible because of the administrative effort needed to install the software on more computers.The time needed for one complete execution of the distributed multiple object tracking algorithm in the lab took 15ms including the time for Ethernet communications.

Accuracy

The error of the position estimates of the CPFA implementation and the software implementation was in the same order of magnitude as the error of the MatLab model in the RoboCup test scenario. The impact of the addition of the negligible likelihood on the accuracy of the CPFA implementation to approximate the true position of the object, was negligible.
In the airport test scenario the system had sometimes problems integrating the very accurate simulated UMMR sensor data. The reason is the fixed point number representation, causing all particles to have the same weight, even when the probability of the samples is slightly different. Thus the importance sampling failed because of the missing gradient.

Resource Usage

The CPFA implementation needs more resources than than the NIOS II/e. Since the NIOS II/e uses no DSP Elements the increase has been estimated by comparing it to a CPFA implementation that also uses no DSP Elements. The CPFA implementation requires between 6 to 19 times the resources of the NIOS II/e implementation. The total required resources of the CPFA can be estimated to be around 7.5 times the amount of resources used by the NIOS II/e. The I7 3537U has 1,400,000,000 transistors according to data sheets from Intel [51], but these are hardy comparable to the logical elements of the Cyclone III FPGA. In the FPGA only a fraction of the transistors perform the implemented logic, while most of the transistors support the logic implementation.

Operation/System	NIOS II/e	CPFA	Increase Factor
DSP Elements	0	98	-
DSP 9x9	0	0	-
DSP 18x18	0	49	-
LC Combinationals	929	9642 (17849)	10.3 (19.2)
LC Registers	510	6887	13.5
Memory Bits	9216	55472	6.0
Total	10655	72001(80208)	6.76 (7.5)

Table 4.4: Resource usage of hardware and software tracker implementations. The numbers in parenthesis denote the CPFA implementation without DSP-Elements. Increase Factor denotes the additional resources needed by the CPFA implementation when compared to the NIOS II/e.

Power Consumption and Heat Dissipation

A simulation analysis showed that the particle filter logic itself consumes about under full load 61 mW. Computationally similar complex tasks require between 300 mW - 900 mW [8] when executed on embedded processors like those used in mobile phones. So GPP's require between 5 - 15 times more power. However the processors used in [8] may not be able to reach the 2 KHz tracker update rate possible with the CPFA implementation under full load. Given that the CPFA is app. 3000 times faster than the NIOS II/e but requires about 7.5 times more resources, an implementation on a NIOS II/e would require

about 400 times more power to process the same amount of data in the same time. The Intel I7 3537U requires approximately 1.3 W of additional energy during the execution of the particle filter algorithm software, which complies to the findings of the authors of [8]. This dynamic power is in addition to the static power consumption of the I7 3757U of 17 W at 2.0 GHz thermal design power (TDP) according to Intel[51].

4.3.4 Performance Comparison

The accuracy is similar for all tested solutions. The I7 3757U is the fastest option, but also the one using the most power and the most resources. The NIOS II/e is to slow. Performance estimates from Altera indicates that the 2 other NIOS II variants would be also to slow, when they are implemented on the same Cyclone III as the NIOS II/e and the CPFA Implementation. The software solution is also the most flexible solution as far as modifications of the particle filter algorithm are concerned. From a first look on the results a software implementation on the I7 or a similar processor seems to be a good solution, but the results of the prototype test support the usage of the CPFA implementation nevertheless. First the software solution does almost fully load 1 core of the I7. The use of the CPFA implementation does not only free the CPUs of a system requiring tracking capabilities from computational load. It also reduces the power consumption also significantly, which is a benefit for all systems with limited power supply. The CPFA implementation does not produce a significant amount of heat and does not need any kind of additional cooling[8] which enables tracking systems to be smaller, more robust, and more energy efficient. Also the speed up advantage of the I7 is not that significant. The software particle filter on the I7 is only 2-3 times faster than the CPFA implementation, while the power consumption of the CPFA implementation is about 20 times lower. By using a bigger FPGA more sample processing units can be used in the CPFA implementation. Since FPGAs are available with speeds and LE capabilities more than twenty times of that of the one used for the prototype, the number of samples and the speed can be increased significantly. Also the basic algorithm of the particle filter does not need to completely flexible for a vast range of tracking scenarios. The ability to adopt the particle filter to application scenario constraints depends mainly on the ability to scale between serial and concurrent execution of the algorithm and the accuracy needed in the application scenario. These characteristics directly determine the LE and memory block usage of the particle filter. The FPGA for the implementation should also contain integrated memory blocks to store the samples. If there would be no memory blocks the samples would require a huge amount of LE. Since the prototype realization fits into a very small, low-speed, and economically priced FPGA, better performing prototypes can be realized using more expensive FPGAs providing a higher speed and more programmable logic resources. Given the estimated speed of the CPFA implementation it should be possible to extend the CPFA to handle multiple tracks. In order to do so the sample storage must be used as a cache for the sample sets of the tracks that are stored in a RAM connected to the FPGA. In this way it is possible to store the state of multiple tracks in the RAM and track a number of objects that is limited by the sensor observation update rate.
Based on these findings a migration of the CPFA implementation from a FPGA to an

[8]The prototype has been run in a permanent loop for days on a development board without the need for a heat sink.

ASIC that acts as a tracking coprocessor could be considered interesting, as the migration is expected to reduce the power consumption of the CPFA about 50% while increasing the processing speed.

4.4 Summary

In this chapter mathematical models and adaptions of the particle filter algorithm that allow an architecture for an efficient concurrent hardware based particle filter are introduced. The soundness of the adaptions and the underlying mathematical models is discussed, and their appropriateness for the task is shown by means of error analysis and comparison to straight forward methods. The performance of the prototype implementation shows that the developed hardware achieves, together with the developed software, the research objectives related to embedded tracking systems. A part of the results is already published [98] and contributes to the state of research.

Chapter 5

Airport Research Objectives

Contents

Initial research on this topic started in as part of the RollMops project that is part of the WFF Project of the BFMI. The project aimed to increase runway safety through the use of local sensors and new embedded data fusion algorithms. In this context the following objectives shall be reached.

1. Show the feasibility of runway incursion prevention systems based on localized surveillance.
2. Develop a design for an experimental runway incursion alerting system (XL-RIAS) based on local sensors and signals.
3. Realize a prototype of the system design using the HWPPTS.

At first glance may seem trivial to design a tracking system capable of monitoring the ground traffic at airports with sufficient accuracy to detect runway incursions in time. A closer look reveals that the circumstances encountered in the airport environment strongly restrict the space of possible solutions. Therefore, in order to reach the above research objectives, first a feasibility study was made that determined the possibilities to apply state of the art sensors and algorithms to the problem and determine configurations that satisfy theoretically desirable performance requirements. The results showed that it is most likely possible to reach the desired accuracy and speed that determine the lower bound of the runway incursion detection and alerting performance, with state of the art hardware and software [97]. Than an extensive survey of approaches to reduce runway incursions was made, see [99]. The performance of these approaches has been compared to the estimated performance of the developed system, and it has been shown that the system outperforms currently deployed technological solutions in some relevant aspects and is able to handle runway incursion scenarios that cannot be handled by the deployed systems. Concurrently the design of the new system has been developed, under the following constraints.

1. The system has to fulfill at least SIL 1 requirements.
2. The system needs to be integrated in existing infrastructure with minimal effort as a standalone solution or to enhance the performance of already installed surveillance systems.
3. The system should be made from of-the-shelf hardware components, as far as possible.

For a better readability the chapter is not organized to reflect the different task involved and the time line of the project but to provide systematic introduction into the topic as a basis and then discuss the technical details of the system.
The chapter is thus organized as follows: The chapter starts with an introduction to runway incursion systems in Section 5.1. Section 5.2 covers the theory of runway incursion prevention systems and their embedding in the airport environment. Technological responses to the runway incursion, including the solution developed in this work, tread are introduced in Section 5.3, and their performance is compared based on the results of a simulation study in Section 5.4. The rest of the chapter deals with the design process[1], the design of the system, the architecture of the system and its integration in the airport environment as well as with the R&D prototype implementation. The chapter finishes with a concise summary.

5.1 Introduction

Runway incursions are occurrences at an aerodrome that involve the incorrect presence of an aircraft, a ground vehicle, or a person on the protected area designated for the landing and take-off of aircraft. The growing traffic volume has kept avoiding runway incursions on the National Transportation Safety Board (NTSB) "Most Wanted" list for safety improvements for over a decade [82]. In the past, runway incursions have led to accidents with significant loss of life. The worst runway incursion accident was at Tenerife, Canary Islands, Spain, in 1997, where two Boeing 747 collided. Recent incidents [90, 81] show that runway incursions are still a problem. Although the number of runway incursions that result in an accident is small, the number of runway incursions has not significantly declined over the last decade. Statistics and results from simulation studies strongly indicate that the number of runway incursions increases much more rapidly than the traffic volume. Depending on the airport topography, an increase of 20% traffic volume may result in a 140% increase of runway incursion potential for a single runway [52].

5.1.1 Definition of Runway Incursions

The International Civil Aviation Organization (ICAO) definition that has been adopted by the Federal Aviation Administration (FAA) in 2007 [33] of a runway incursion is the following [49]:

"Any occurrence at an aerodrome involving the incorrect presence of an aircraft, vehicle, or person on the protected area of a surface designated for the landing and take-off of aircraft."

Runway incursions are classified in relation to the severity of the incident, with classifications A - E, as shown in Table 1. The most relevant classifications for Runway Incursion Prevention and Alerting Systems (RIPAS) are A and B, where time is critical.

[1]Since the system is safety related the design process had to be modified in such a way that it requires a more detailed explanation.

Severity classification	Description
A	A serious incident in which a collision is narrowly avoided.
B	An incident in which separation decreases and there is significant potential for collision which may result in a time-critical corrective/evasive response to avoid a collision.
C	An incident characterized by ample time and/or distance to avoid a collision
D	An incident that meets the definition of runway incursion, such as the incorrect presence of a single vehicle, person or aircraft on the protected area on a surface designated for the landing and take-off of aircraft but with no immediate safety consequences.
E	Insufficient information or inconclusive or conflicting evidence precludes a severity assessment.

Table 5.1: ICAO severity classification scheme [49]

Classifications C and D should also be detected by RIPAS but do not require immediate action.

5.1.2 Causes for Runway Incursions

The FAA and ICAO both classify runway incursions according to the following types of causes that lead to the incursion: "Pilot Deviations" (PD), "Operational Errors/Deviations" (OE), and "Vehicle/Pedestrian Deviation" (VPD).

Pilot Deviations

Pilot deviations are actions of a pilot that violate any Federal Aviation Regulation. For example, a pilot deviation occurs when a pilot crosses a taxiway hold line, entering a runway for which the aircraft has not been authorized by an air traffic controller (ATCO) to enter. For a detailed description of such a case, see the reports [90, 81] of the NTSB and the German Federal Bureau of Aircraft Accidents Investigation (BFU).

Operational Errors / Deviations

Operational errors / deviations are actions of ATCOs that result in either less than the minimum separation between two or more aircraft and vehicles or an aircraft landing or departing on a runway that is closed to the aircraft. For example, an ATCO could commission an aircraft to land on a runway that is already in use. The report in [84] provides good insight into such a situation.

Vehicle / Pedestrian Deviations

Vehicle / pedestrian deviations include pedestrians, vehicles, or other objects that interfere with aircraft operations by movements that have not been authorized by air traffic control (ATC) and/or APRON controllers. A serious runway incursion occurs when, for example,

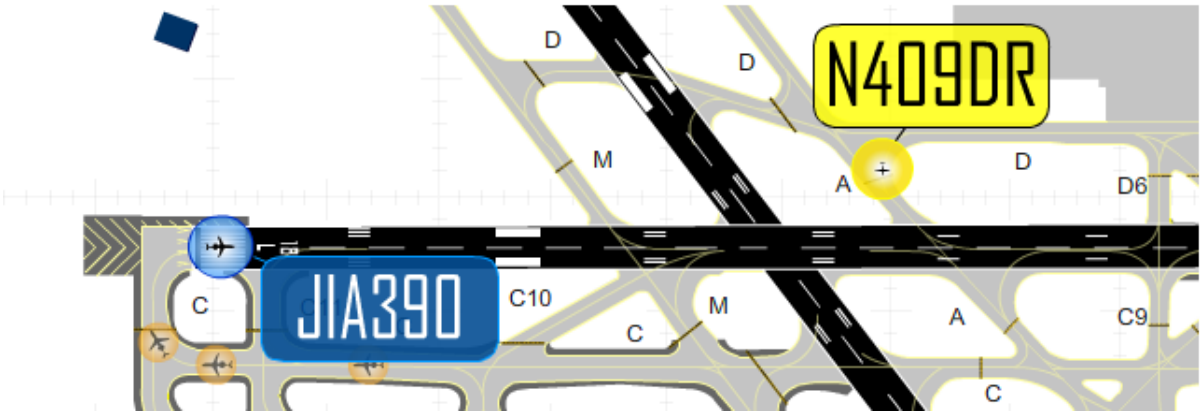


Figure 5.1: Initial situation of the category A incident that happened at Charlotte Douglas International airport on May 29, 2009.

service technicians enter a runway unauthorized, a taxiing Boeing 747 nearly causes a collision, or when a service car left on the runway collides with a landing aircraft.

5.1.3 Runway Incursion Examples

For a more vivid impression two examples of runway incursions are given, both representing the type of incursion that is particularly targeted by the RollMops project. The fist incident was reported to the FAA and conditioned by the FAA, it happened on May 29, 2009. For the details of the incident refer to [81]. The second incident was examined by the German Federal Bureau of Aircraft Accidents Investigation (BFU). Both incidents can be classified as category A, a real dangerous situation. The first incident was the result of an operational error (OE), the second incident was caused by a pilot deviation (PD).

ADSE-X

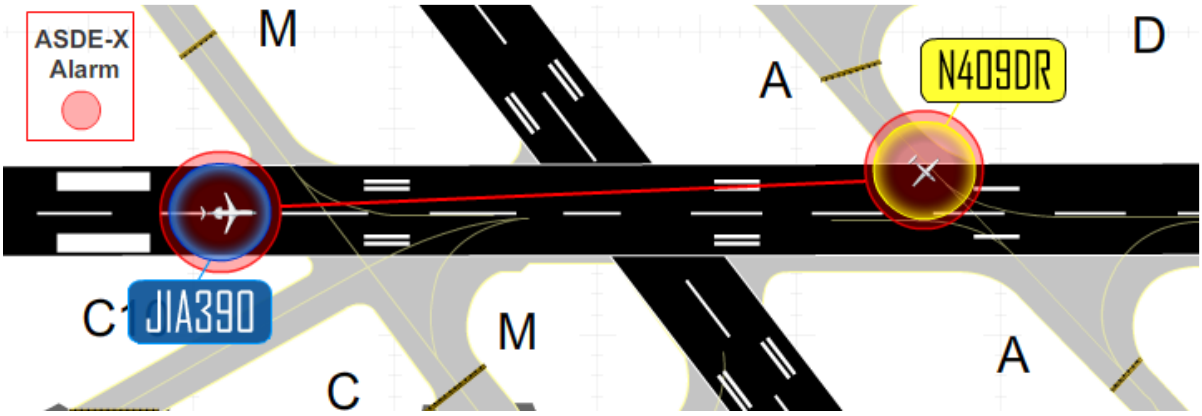


Figure 5.2: Situation at the time of alert of the category A incident that happened at Charlotte Douglas International airport on May 29, 2009.

The initial situation is shown in Figure 5.1. Flight JIA390 a commercial airliner is cleared by local control east (LCE) to take off from Runway 18L full length. At the same time a general aviation aircraft Flight N409DR was cleared by LCE to taxi into position and hold on Runway 18L.
Apparently LCE was not aware that flight N409DR was not lined up behind Flight JIA390 but rather planned to begin its take off roll from the intersection downfield. Flight JIA390 begun its take off roll while Flight N409DR was proceeding to its holding position.
When Flight N409DR entered the runway, Flight JIA309 had already rolled about 1600 feet on the runway and had accelerated to approximately 227 feet per second. The situation is shown in Figure 5.2. At that time the ASDE-X system deployed on the airport raised an alarm. It took a few seconds for LCE to communicate the alert to Flight JIA309 which had already spotted Flight N409DR and initialized an evasive maneuver while decelerating. Flight N409DR became aware of the situation a few seconds later and began an evasive action.
Figure 5.3 gives a good impression of the outcome of the incident. Booth aircraft came to halt with a horizontal separation of approximately 10 feet. The full recording of the incident can be found at http://www.faa.gov/airports/runway_safety/videos/.

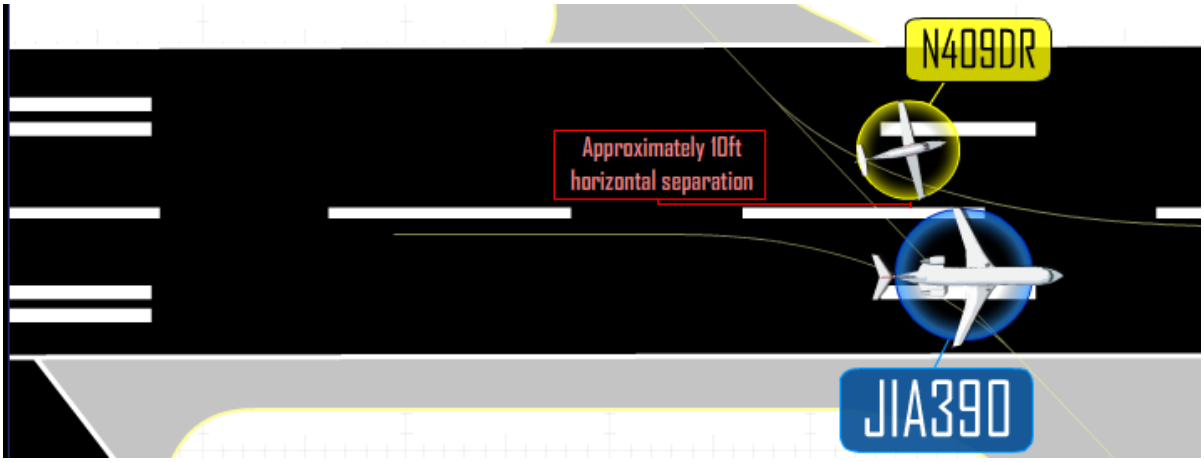


Figure 5.3: Outcome of the category A incident that happened at Charlotte Douglas International airport on May 29, 2009.

BFU

This incident happened on May 3, 2004 at the Munich International Airport. The involved parties were air traffic control (ATC) and two commercial air liners. The incident was investigated by the German Federal Bureau of Aircraft Accidents Investigation and an investigation report was published in February 2009 [90].
The situation was caused due to a misinterpretation of the situation by the pilots of Flight AT45 that mistook a departing flight A321 as the landing flight B733 it was supposed line up behind. Figure 5.4 illustrates the situation. Initially Flight AT45 had clearance to line up behind the next landing and Flight A321 was cleared for take off. Flight A321 took of and was mistaken by Flight AT45 as the landing flight it was supposed to line up behind. The left image in Figure 5.4 shows the situation during the take off roll of Flight A321; the right image shows the situation a few seconds later when the landing Flight

B733 passed the runway threshold. Flight AT45 has already moved some distance and is proceeding towards the runway.

Figure 5.4: Inital situation, radar images from [90].

Flight B733 became aware of the dangerous situation immediately after touchdown on the runway. The left image in Figure 5.5 shows the radar print of the situation at touchdown; Flight AT45 is still about 40m from the runway centerline. The right image shows the situation when Flight B733 became aware of the situation. The pilots used reverse trust and max. auto brake to steer clear of Flight AT45 which was still unaware of the situation.

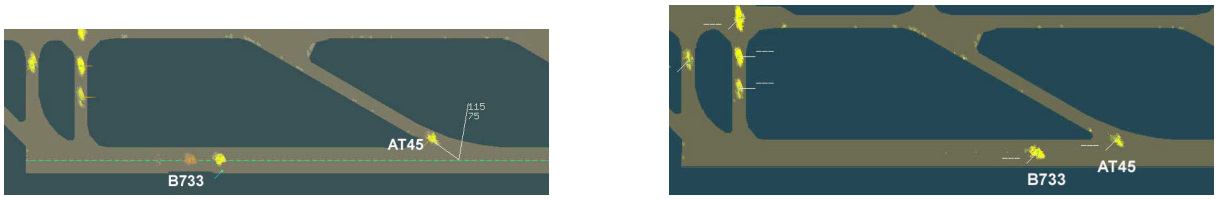


Figure 5.5: Development of the initial situation , radar images from [90].

A few seconds later Flight B773 passed Flight AT45 at a distance of a few meters with a speed of approximately 110 knots / 205 kmh with a horizontal separation of a few meters as perceived by the crews of the two aircraft. A radar print is given in Figure 5.6. The left image of the print shows the most dangerous situation when Flight B733 passed Flight AT45. The right print shows the situation after the airplanes have passed.

Figure 5.6: Near collision and outcome of the incident, radar images from [90].

Interestingly the Runway Incursion Monitor (RIM), a part of the Airport Surface Movement Radar (ASMR), did raise an alarm related to the incident but the notification was turned of at the workstation of the air traffic controller. The reason was that the system did generate to many false alarms.

5.1.4 The Role of General Aviation

General aviation (GA) accounts for the majority of runway incursion incidents, including those that involve air carrier operations. Air carriers and large airports are usually better

suited to be outfitted with technology that is related to runway incursion prevention. The technical and financial constraints for most of the general aviation aircraft and smaller airports limit the use of runway incursion prevention technology. In particular, the use of transponder systems and electronic flight bag (EFB) equipment poses a problem, and this will be discussed in Section 5.3.

5.1.5 Runway Incursion Hot Spots

Runway incursions occur on hot spots, which are defined by the ICAO as "A location on an aerodrome movement area with a history or potential risk of collision or runway incursions and where heightened attention by pilots/drivers is necessary." [49]. To avoid runway incursions, the ICAO demands that explicit runway crossing clearance is required to cross holding positions. Typical examples of hot spots are shown in Figure 5.1.5. Clearance usually is given via radio telephony (RTF) and, if applicable, by signals at hold lines.
For several reasons, a common cause for runway incursions is that aircraft proceed beyond their holding position onto the runway without clearance or are cleared to enter a runway that is already in use. These scenarios are among the most dangerous situations because a plane on the runway that is too slow to take off and too fast to stop has almost no option to respond to the situation, an opinion shared by other studies [49, 9]. Currently deployed runway incursion detection systems need in between 2 and 6 seconds to detect such a situation. An earlier detection would enable warning the vehicle on the taxiway, which, because of its slower speed, is better suited to react to a situation during its early stages of development.

5.1.6 Runway Incursion Statistics

In Europe, 600 runway incursions were reported in 2005, but the European Organization for the Safety of Air Navigation (EUROCONTROL) stated that the reporting culture is ineffective and that not many reports are made because of judicial and cultural barriers. The number of reported runway incursions increased in the EUROCONTROL Statistical Reference Area (ESRA) to nearly 1000 in 2008 [30, 31] . This increase partly reflects a better reporting culture[32], e.g., the "Runway Safety Report 2008" of Zurich airport [120] provides a detailed analysis of the runway incursion developments at the airport since 2000. Notable is the high number of category A and B incidents in ESRA, which reached a peak in 2006 at 63 incidents; the annual number of incidents has not changed much since 2003. For major airports (e.g., Heathrow), up to 30 runway incursions per year have been reported [16]. For the USA, more detailed information is available, which is summarized in Table 5.2. Between 2003 and 2010, the FAA registered a total of 6.989 Runway incursions. Apparently, these incursions are only for towered airports; incidents that were classified as runway incursions at untowered airports from the National Aeronautics and Space Administration (NASA) and NTSB database cannot be found in the FAA database [15]. For this total number, only category A and B incidents are considered to have a significant potential of evolving into an accident. According to the FAA, the number of runway incursions of category A and B have been decreasing since 2003. Most notable is the significant decrease of these events in 2009 and 2010, which is a trend that the FAA

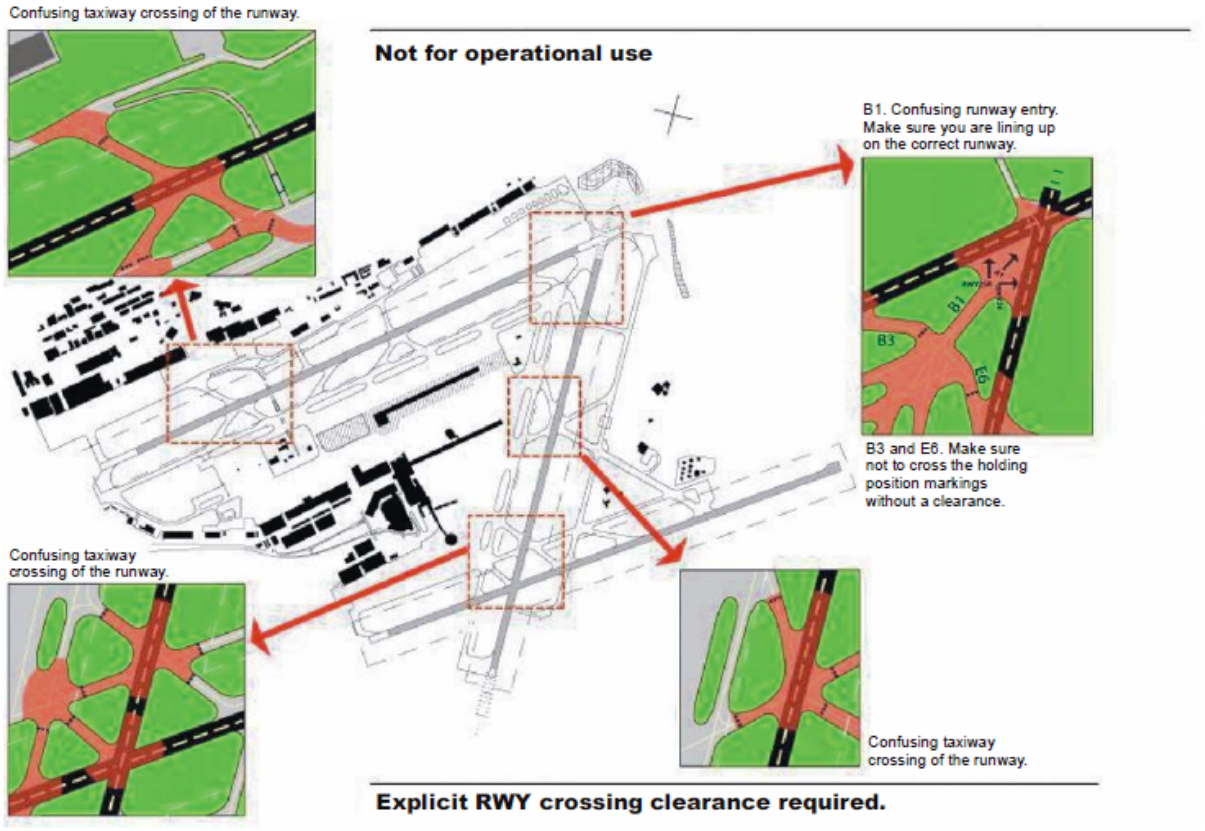


Figure 5.7: Hot spot examples from ICAO manual on the prevention of runway incursions[49]; the map shows hot spot locations at an airport. For each hot spot, advice for pilots is provided. Confusing taxiway crossings of the runway at 3 hot spots cause pilots to stop on the runway in search of the taxiway or to miss the taxiway, forcing them to travel on the runway. The difficult topography at the northern hot spot may mislead pilots into entering the wrong runway, and the hold line at E6 is so close to the runway that aircraft holding short of the runway beyond the hold line may easily collide with landing aircraft.

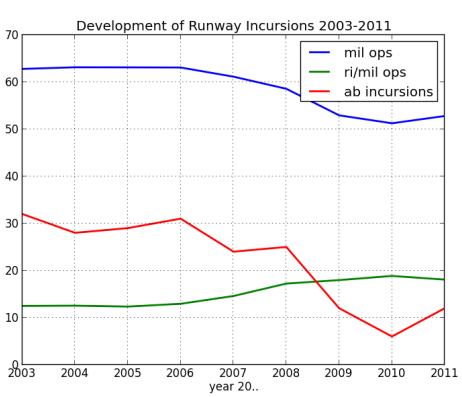
attributes to its program for runway incursion prevention.
A study found that the assignments to categories have changed from 2009 on, and that similar events that have been considered severity category A or B before 2009 are now categorized as C. In a survey of the incidents of 2010, only 5% of 174 pilots, including 75 airline pilots, agreed with the severity that was assigned by the FAA [15]. The assignments given by the pilot survey group would apparently raise the number of A and B incidents to a total number of 28.
The majority of runway incursions results from PD and VPD. However, a closer look at the FAA statistics reveals that the OE and PD make up the major part of the serious runway incursions of types A and B (see Figure 5.1.6). Therefore, runway incursion prevention technology should address PD and OE with priority. The ratio of runway incursions that result in a collision can be estimated at 1 collision in 300 runway incursions [120].

Year	RI-FAA	RI/M.Ops	A,B	A,B Comm.	Total Ops(arir06)
2003	784[2](323)	12.5(5.1)	32	10	(62.783.048)
2004	792[2](326)	12.5(5.2)	28	9	(63.124.797)
2005	779(327)	12.3(5.2)	29	9	63.104.415 (63.108.846)
2006	816(330)	13.4(5.4)	31	10	61.076.341(61.334.693)
2007	892(367[1])	14.6(6.0[4])	24	8	61.133.748
2008	1009(415[1])	17.2(7.0[4])	25	9	58.562.343
2009	951(391[1])	17.9(7.3)/14.9(6.1[4])	12	2	52.928.316[5]/63.774.300[3]
2010	966(397[1])	18.9(7.75)/13.9(5.7[4])	6	?	51.249.476[5]/69.450.300[3]
Total	6.989	181(187)	57[5]		503.009.100

Table 5.2: FAA runway incursions overview. The second column lists the total number of runway incursion incidents, the numbers in brackets refer to the old FAA runway incursion definition. The third column shows the ratio of runway incursion per million operations. The fourth column lists the total number of category A and B incidents. The fifth column lists the total number of category A and B incidents involving commercial airliners. The last column contains the total number of airport landing and take-off operations.[1] Estimate based on the number of FAA incidents. [2]Estimate based on the number of incidents according to an old FAA definition (before 2004). [3]Estimate based on the FAA traffic forecast. [4] Calculated from estimated incidents. [5] New numbers from [84].

5.2 Theory of Runway Incursion Prevention Systems

Runway incursion prevention technology is based on protecting measures against causes that lead to a runway incursion and providing alerts during the cause of a runway incursion. For example, safety logic could prevent Air Traffic Control (ATC) from commissioning more than one aircraft to use the same runway, thus providing protection against this type of Operational Error (OE). To achieve this runway incursion protection requires removing the human from the loop as much as possible. Thus, the general architecture of runway incursion prevention technology looks like the architecture shown in Figure 5.2. The primary input to the system is given by information from various sensors and from traffic information service networks. This information is usually fused by multi sensor data fusion, integrating background information such as maps and movement models into tracks describing the movements at the airport. This description is evaluated, and ATC commands such as route information are integrated to assess the traffic situation, to predict conflicts and to detect runway incursions. Information about the traffic situation is given to ATC, pilots and vehicles via a human machine interface (HMI), e.g., an Electronic Flight Bag (EFB) if available, and signals at the airport or via radio telephony (RTF) are from an Air Traffic Controller (ATCO). The fact that the architecture communication of the distributed components belongs to the technology is important because a communication infrastructure to support high speed data transfers from/to sensors and signals distributed across the airport is not always available. For example, the operation of intelligent signals on serial circuits requires the use of power line communication technology with sophisticated algorithms to ensure real-time constraint compliance.



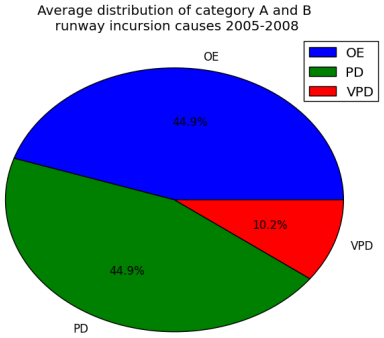


Figure 5.8: Runway incursion types in numbers and as type ratios. For runway incursion prevention technology, the distribution among categories A and B are of interest. Operational errors and pilot deviations contribute substantially to the most dangerous types of runway incursions, while vehicle/pedestrian deviations result in only a few serious incidents.

5.2.1 The Importance of Situational Awareness

Previous and current studies on this topic agree that situational awareness is a key to runway incursion avoidance and the safe handling of runway incursions. Almost no flight crew intentionally taxis active runways. In [119], the authors estimate a reduction rate of 80% from experiments to be reachable by enhancing the situational awareness of flight crews and ATCs. Current installations of Final Approach Runway Occupancy Signal (FAROS) and Runway Status Lights (RWSL) have been found to reduce runway incursions by 70% at the Dallas/Fort Worth International Airport (DFW). The success of the installation at DFW, even though it is below the estimated reduction rate from the National Aeronautics and Space Administration (NASA) study [119], is likely related to its ability to help flight crews avoid runway incursions. Basically, there are two approaches to prevent runway incursions: avoiding entering a runway in use and detecting an imminent runway incursion as early as possible and resolve the situation. The abilities to prevent a runway incursion directly depend on the surveillance system that provides the input for the systems or persons dealing with this problem. The importance of a reliable, effective and accurate surveillance system with a coverage of the maneuvering area of an airport cannot be understated; this opinion is supported by other research reports [119, 102, 103, 48, 104]. The integration of Runway Incursion Prevention and Alerting Systems (RIPAS) into the work flow of the ATC also depends on the reliability of the surveillance technology [104]. In the following, the approaches and their dependency on the accuracy of the surveillance system is discussed.

Runway Incursion Avoidance

The ability of a flight crew to avoid inadvertently entering an active runway is called avoidance. Young and Jones [119] distinguish between 3 influential factors: Own Ship Position Awareness, Route Awareness and Route Deviation Detection. Other authors also include Runway Activity Status as an influential key factor [49, 102]. For a better

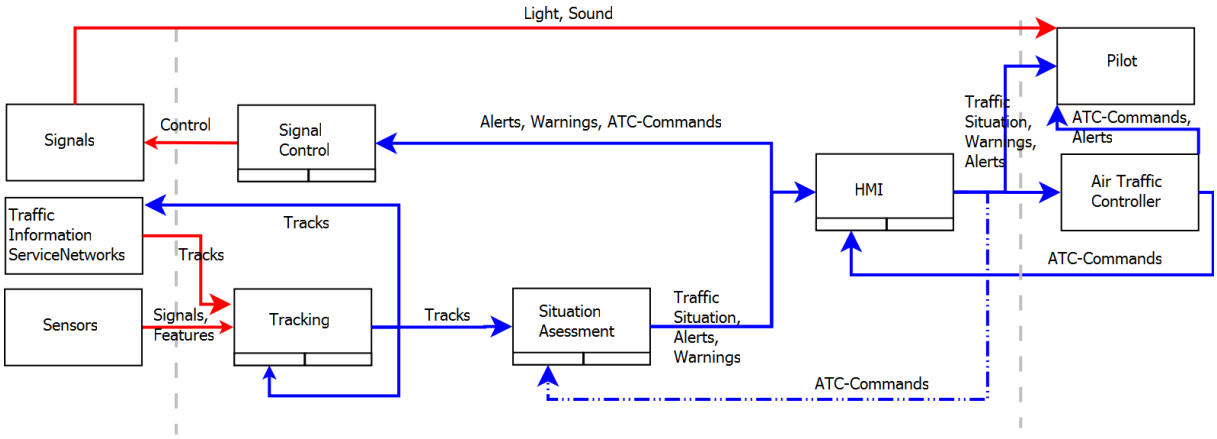


Figure 5.9: The basic formal architecture of a runway incursion prevention system. The primary input to the system is provided by sensors and traffic information services and is input from Air Traffic Controller (ATCO) and pilots via a Human Machine Interface (HMI). From this information, the system assesses the traffic situation and provides warnings or alerts to pilots and ATCOs via airfield lighting signals or HMIs.

understanding of the downstream text, a short description of these terms is provided.

1. Own Ship Position Awareness: The ability of a flight crew to estimate the position of their own aircraft on the aerodrome, for vehicles without the means to determine their position directly via Global Positioning System (GPS) and Map; an accurate surveillance system enables ATCs to inform traffic participants of their position.

2. Route Awareness: The ability of a flight crew to map the routing instructions given by ATCs to the topography of the aerodrome. With sufficiently accurate surveillance, routing information could be directly and automatically given to the pilots, via visual inputs, i.e., following the green paradigm.

3. Route Deviation Detection: A frequent cause for runway incursions is the deviation of aircraft from their designated route. ATCs can inform flight crews if they leave their designated route. Because of the topology of the airports, the specified requirements appear to be sufficient at first glance; however, a currently deployed system could detect a vehicle leaving a taxiway and crossing a runway at a high speed but not as fast as desired.

4. Runway Activity Status: Runway status signals that show the activity status of the runway work much like traffic lights for road traffic. If the status of a runway is clearly visible to traffic participants, then they are unlikely to enter a runway that is designated for use for another aircraft, even if they do not know where they are. This feature does not depend heavily on the accuracy of the surveillance if the signals are set by ATCs as soon as the runway is designated to another aircraft. If the signals are set only by the detection of runway traffic, then accuracy becomes more important for preventing conflicting situations.

Runway Incursion Detection

The ability to detect a runway incursion enables the involved parties to respond accordingly and to prevent, or at least minimize, the damage. Three key factors appear to contribute to the ability to detect a runway incursion.

1. Traffic Position Awareness: The accuracy and delay of the surveillance do directly determine the quality of the traffic position awareness. This consideration is important because the accuracy of the traffic positions directly affects the delay from the incursion to the alert. During their research, the developers of the Runway Incursion Prevention System (RIPS) found that, in scenarios that involve a landing plane [119], the six second delay needed to inform the pilots of the landing aircraft could be the time needed to prevent a fraction of the accidents.

2. Own Ship Position Awareness: The ability of a flight crew to estimate the position of their own aircraft allows it to notice that they are about to enter an active runway, or that the runway they are on has just been designated to another plane, given that the flight crew has access to the runway activity status. If their own position awareness and traffic position awareness is good, then an imminent runway incursion can be detected easily. The accuracy of the surveillance determines the ability of the flight crews to observe the actions of other traffic participants, i.e., to determine whether a plane is holding short of a runway or entering the runway.

3. Runway Activity Status: Because the definition of a runway incursion is based on the activity of the runway, the activity status of a runway is very important information. Regarding surveillance accuracy, the same constraints apply as for runway incursion avoidance.

5.2.2 Good Runway Incursion Technology

RIPAS should enhance own position awareness, route awareness and traffic position awareness of both ATC and traffic participants. The runway activity status should be known to all of the traffic participants and the ATC. RIPAS should react immediately to route deviations and runway incursions and provide direct and immediate information to both the flight crew and the controllers [119, 102, 103]. If RIPAS satisfies these requirements, then the following would happen:

1. Decrease the probabilities for events leading to a runway incursion, thus leading to a reduced runway incursion ratio.

2. Decrease the severity of the outcome of runway incursions and prevent accidents

The general idea behind this assumption is that the traffic depends on a series of decisions by humans. Both false and insufficient information can lead to poor decisions. Figure 5.10 shows how RIPAS affects the safety of airport operations. By providing reliable sufficient information by means of technology, poor decisions will be minimized. The remaining poor decisions will be noticed by surveillance and control technology and, depending on the situation after a poor decision, a warning or an alert will be issued. RIPAS technology can be installed in vehicles, aircraft, and at the airport. Many different technologies by

Figure 5.10: A simplified view on the impact of RIPAS technology on airport operations. False or insufficient information (in gray) leads to poor decisions (such as crossing a closed hold line), which result in dangerous operations. RIPAS attempts to provide reliable information that is sufficient for good decisions (such as showing the status of a hold line by signals). The remaining poor decisions should be detected by RIPAS, and a warning should be issued to prevent dangerous operations. If dangerous operations occur anyway, then the system shall issue an alert so that the operation can be canceled and a status of safe operation can be obtained.

different vendors can operate in parallel to maximize the safety net effect. Thus, RIPAS should provide a standard interface to communicate its data to other RIPA Systems.
A rather abstract mathematical model of the influence of such technology is shown in Figure 5.11. Here the operation of the airport is modeled using a Markov model. Most time the airport is in the mode safe ground traffic, but there is a slight chance of an imminent runway incursion that may slip the notice of the involved parties. This imminent runway incursion develops to a runway incursion.

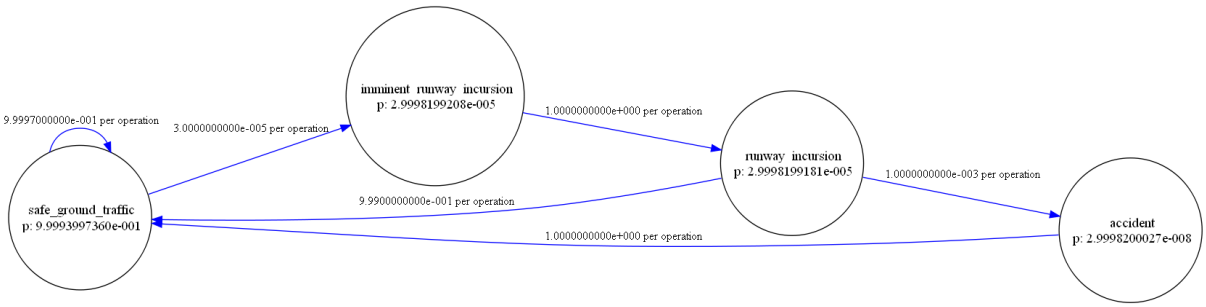


Figure 5.11: Airport operation without RIPAS.

Most runway incursions resolve into an incident without consequences and the airport returns to the safe ground traffic mode. In the unlucky case the runway incursion results in an accident, nevertheless the airport resumes normal operation afterward. The equilibrium probabilities are given per operation in this Markov model, and have been computed with an arbitrary precision number libraries. Probabilities per hour or year can be calculated from these based on the number of operations per year. For major airports this may result in between 10-30 runway incursions.
The model in Figure 5.12 shows the influence of a RIPAS. Transition probabilities are the same, but two new nodes and some new edges are in the model. The RIPAS influence begins when a runway incursion is imminent. The RIPAS detects the imminent runway incursion with a high probability. For the detection probability SIL 1 is assumed for the RIPAS which is not met by any currently deployed system but is a desirable reference.

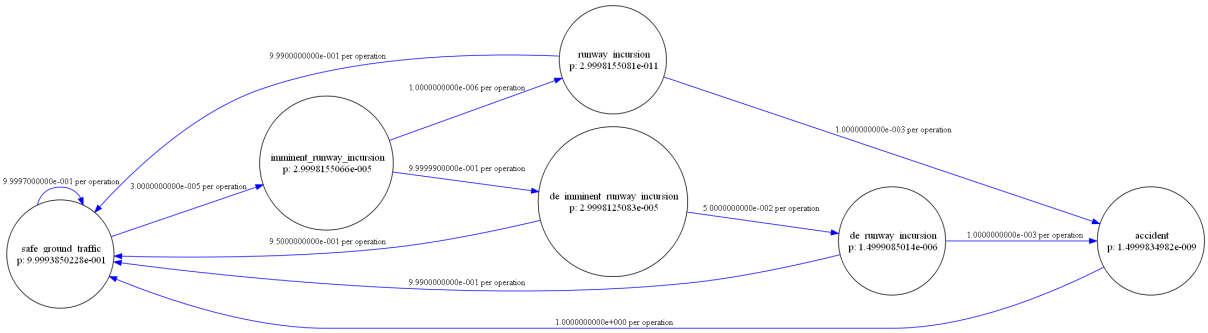


Figure 5.12: Airport operation with RIPAS.

If an imminent runway incursion is detected the alerting functionality of the system will very likely enable the involved parties to resolve the situation without an accident. The likelihood for preventing a runway incursion based on alerting systems has been estimated from NASA studies. Even if the situation cannot be resolved and develops into a runway incursion, the involved parties are much more aware of the dangerous situation which will result in a better ability to react accordingly. So even at this stage there is a high probability to resolve the situation and prevent an accident. The Markov model analysis show that the probability for an accident is 20 time lower compared to the model without a RIPAS. The probability of an undetected runway incursion is about 10^6 times lower than without a RIPAS. The model can be refined to reflect OE, PD, and VPD occurrences and local circumstances at individual airports, but this was not done because of time constraints on the project.
Based on these model one could argue that a RIPAS that only detects half of the imminent runway incursions would also increase the safety of the airport significantly and therefore there is no need to make a RIPAS very reliable. This is not the case because unreliable technology is turned of or ignored by the people running the airports as well as by pilots. Both parties require reliable equipment to work with and have refused to work with unreliable systems in the past. There are several runway incursion incidents where unreliable RIPAS where turned off because of the high rate of false alarms.
The probabilities show another problem. The HMI must be very clear in its meaning if there is a runway incursion, because runway incursions do not happen that often in the live of a ATCO or pilot and therefore an unclear HMI message might be ignored. This has happened in several runway incursion incidents in the past, where RIPAS issued an alert, but the personnel was not able to understand the meaning of the rather abstract alerting signal.

5.2.3 Runway Incursion Prevention Key Technologies

To realize a system with the general formal architecture or a part of it, key technologies are used. These provide the basic components from which RIPAS can be build. Often proven-in-use equipment is the first choice when new RIPAS are developed. The following text introduces the most important key technologies related to runway incursion alerting and prevention, especially surveillance sensors, traffic information networks, human machine interfaces and algorithms for tracking vehicles and assessing the traffic situation, including the detection of runway incursions.

Surveillance Sensors

Surveillance sensors provide information about their environment, which is usually preprocessed into a compact meaningful representation by the sensor. They can be part of an airport infrastructure or part of an aircraft and/or other mobile units. Not all currently available sensors are equally suited for runway incursion prevention. With respect to airport traffic surveillance, cooperative and non-cooperative sensor technologies can usually be distinguished. Cooperative sensor technology requires that a part of the technology be installed in the aircraft or vehicle that the sensor is measuring, for example, active transponders. In contrast, non-cooperative sensor technology does not need any part on the other side but instead usually requires more complex and often less robust methods to provide meaningful information. Global sensors cover most of the area that is under surveillance while localized sensors measure only a part of the area. As a result, localized sensors usually provide more accurate information at a high update rate, while information from global sensors is less accurate and has a slower update rate. However, localized sensors have the drawback that, unless they have their own tracking and situation assessment functionality on board, they need to transmit the sensor data to ATC for further processing, which requires up to many kilometers of new cables. Limitations in the bandwidth of power line communications usually do not allow the sensors to transmit all of the sensor data via a power line, especially if there is more than one sensor.

Traffic Information Service Networks

Traffic information service networks (TISN) provide information on the local traffic situation. This information can be the position of an aircraft that tracks its own movement via GPS and is sometimes supported by inertial sensors or tracks from an airport traffic surveillance system. A runway incursion prevention technology based only on TISN has the drawback that it does not consider mobiles that do not participate in the network, unless they are under the surveillance of an airport participating in TISN. Additionally, TISN could suffer from communication errors or might be entirely shut down for security reasons.

Tracking Algorithms

Tracking algorithms integrate information from different sources, such as sensors and TISN, to generate tracks that represent an aircraft or vehicle. Filtering provides a means of interpolating or predicting the state of the tracks in the time between the sensor updates. Popular tracking algorithms are particle filters and Kalman filters. These methods encompass a wide variety of techniques that are adapted to specific purposes and that have been applied, with great success, to airport traffic surveillance. However, even the best tracking algorithms cannot compensate for inaccurate sensor information when detecting rare events such as runway incursions.

Situation Assessment Algorithms (SAA)

Because of the complexity and the dynamics of airport traffic operations, both the ATC and traffic participants cannot be fully aware of dangerous situations even when they have full information that is provided by a reasonable HMI. Situation assessment algorithms

(SAA) analyze a set of tracks, background information, such as airport maps, and auxiliary information, such as routing information, to assess the traffic situation and to detect possible conflicts such as runway incursions. Situation assessment algorithms can also provide predictions about the usage of runways and taxiways in the near future, supporting ground handling and movement control (GMC) and ATC. SAAs work well as long as the traffic data are accurate, timely, and reliable [10]. However, this scenario is usually not the case, and sensor data are often unreliable, suffering from systematic and random errors that cause the wrong detection of non-existing conflicting traffic situations (false alerts) or a late detection of truly existing runway incursions [119, 10, 110, 53].

Human Machine Interfaces and Signals

All of the information would be of less use if it were not available to the appropriate involved parties. Human machine interfaces, such as intelligent interactive airport maps for ATC, GMC or as part of the EFB for pilots, are well suited for this purpose but often require a retrofit of the cockpit or workstations and a communication infrastructure in the airport. The alternative is to have signals that indicate important information to the traffic participants, e.g., the runway activity status or runway incursion alerts. However, the installation of signals is a costly process. If airports do not want to install kilometers of cable, then existing power lines must be used to communicate with signals.

5.3 Available Systems for Runway Incursion Avoidance and Detection

Most of the currently operating or proposed Runway Incursion Prevention and Alerting Systems (RIPAS) use standard technology components. An overview of the different technological implementations of the functionality that were introduced in the previous section as well as RIPAS systems is given. In the following full scale systems supporting runway prevention are introduced. These systems strongly rely on readily available standard key technologies. Readers unfamiliar with airport technology such as traffic signals, sensors and human machine interfaces like airport surface traffic displays may refer to the Appendix A for details on these systems. Another crucial part of these systems are the underlying algorithms. Most of these algorithms follow the same basic ideas that are explained first before the section continuous with a concise description of full scale RIPAS.

5.3.1 Runway Incursion Detection Algorithms

Runway Incursion Prediction and Detection Algorithms (RIPDA) are algorithms that detect runway incursions in their early stages or predict them before they really happen. In the following text the basic theory of the detection and prediction of runway incursions is introduced.

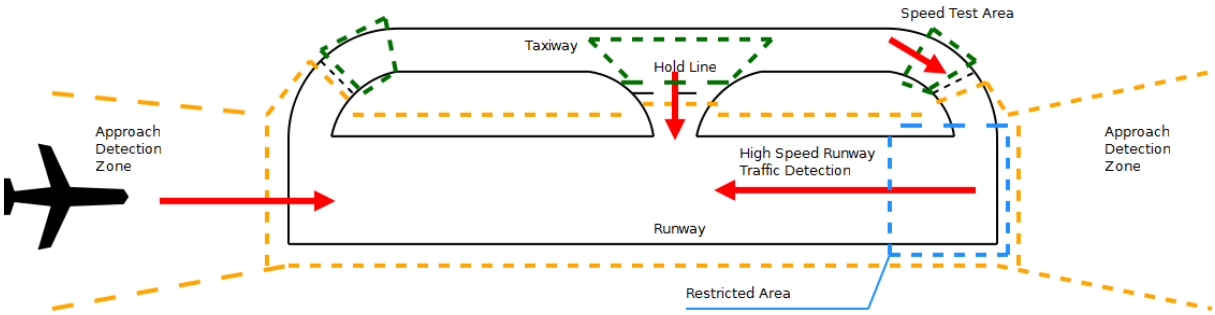


Figure 5.13: Important zones for the generic runway conflict detection algorithm. The most important area is the runway, including the runway entrance area to the hold lines. At speed test areas on the taxiway in front of the hold line, a vehicle is checked for whether it is approaching a closed hold line at high speed. Approach Detection Zones are used to automatically determine whether an aircraft is about to land on a runway.

Generic Runway Conflict Detection:

The generic runway conflict detection approach is to track only traffic participants inside a designated area around a runway. This approach is used by PathProx, Runway Safety Monitor (RSM), and Basic-Runway Safety Net (B-RSN) algorithms. Often dedicated volumes on the airport surface and on the final approach paths are used to determine specific variables needed for the detection of a runway conflict. To clarify the following explanation of the different zones Figure 5.1.5 provides a simplified view of these volumes. Depending on the topographic position the following volumes can be distinguished.

Runway Area: The runway itself is a very important area for runway incursion prediction and detection. Depending on the traffic situation on the runway and in the air, a runway can be active or inactive. It is possible to determine the runway activity status automatically given traffic surveillance equipment with reasonable accuracy.

Final Approach Detection Zones: Final Approach Detection Zones (FADZ) are areas extending from the runway threshold to about 2nm along the corridor used by aircraft for the final approach on runway. They are used to detect aircraft that are about to land on the runway or taking off from the runway.

Speed Test Areas: Runway Incursion Speed Test Areas (RISTA) are areas on the taxiway in front of a hold line. The algorithm tries to predict, based on the estimated speed of the vehicle, if the vehicle is to overrun a hold line/stop bar. To avoid nuisance alerts the algorithm should be informed about the status of the stop bar and/or the activity status of the runway. RISTAs can be used by both cockpit and airport based systems.

Virtual Stop Bars: Virtual Stop Bars are hold lines under surveillance. If a vehicle crosses the line without clearance a stop bar violation is detected. Because estimated positions of vehicles, tracked with primary surveillance equipment, moving at low speeds may jump backwards, just testing for the crossing of a line might result in false alerts.

Algorithms based on secondary surveillance equipment can be used for a more reliable surveillance of virtual stop bars. Virtual stop bars are used mostly by airport based RIPAS.

Restricted Areas: The areas around positions dedicated for aircraft to wait for their take-off roll can be defined as restricted areas. These areas can be used to determine the activity status of the runway or detect traffic without proper clearance. They are particular useful together with digital Air Traffic Control (ATC) routing and clearance information. If such information is available and the tracked vehicle is labeled, either manually or by the use of a cooperative transponder, the algorithm can detect deviations between route, clearance and vehicle position. However manually labeling has been found to be prone to false identification in operational test [53].

Scenario Based Runway Incursion Detection: Prediction and detection can be done based on traffic scenarios e.g.: PathProx distinguishes over 40 runway incursion scenarios. The course of runway incursions often follows a similar pattern. By matching the movement of the traffic near the runway to these patterns it is possible to detect situations that may evolve into a runway incursion and detect runway incursions. This approach allows the use of two stage alerting systems, particular suited for cockpit use. If a traffic situation can evolve into a runway incursion a warning is issued to draw the attention of the pilots to the dangerous situation. In case of a runway incursion an alert is issued. However some of the more serious incursion incidents happen in normal situations; e.g. if an aircraft, that was waiting at a hold line, taxis onto an active runway because of a wrong clearance or a deviation of the pilot. These scenarios can pose a problem because a warning is inappropriate for a normal situation like an aircraft waiting at a hold line next to an active runway.

Runway Activity Status Detection: The activity status of a runway can be determined automatically, manually by ATC or using a mixture of both approaches. The automatic detection has the advantage that it can support Air Traffic Controllers (ATCO) during their work, acting as a safety net. It can also be used to activate stop bars and set signals automatically, e.g. Runway Status Light System (RWLS) and Final Approach Runway Occupancy System (FAROS) with Airport Surface Detection Equipment - Model X (ASDE-X) input switches RWLS and FAROS components according to the runway activity status. Runways become active if they are used for take-off or landing operations, regardless of the status of the runway as supposed by ATC. The automatic detection of the runway activity status works also for wrong runway landings or take offs. Algorithms for the automatic detection of the activity on the runway have been applied with success during ASDE-X and Advanced - Surface Movement Guidance and Control System (A-SMGCS) tests. Because it is not recommended to set the runway activity status to active whenever there is some vehicle on the runway, the following criterion can be used to determine the activity status of a runway.

Final Approach The detection of aircraft on their final approach to a runway is done by checking the path of aircraft that are not on the ground. If the aircraft is within the approach corridor of a runway for some time and the trajectory of the aircraft is

converging on the runway, the algorithm assumes that the aircraft is on its final approach on the runway. The runway activity status of the runway is set to active. This algorithm works for technology on the side of the approaching aircraft as well as for airport based systems.

High Speed Traffic: The runway activity status is set to active if there is traffic on the runway that exceeds a certain speed limit and is traveling along the runway. Setting a runway active when a vehicle is crossing but not traveling along the runway when following its taxi route or during take-off or landing is not recommended. Because non cooperative aircraft do not send their positions via Automatic Depended Surveillance - Broadcast (ADS-B) cockpit based systems may not be able to detect all traffic unless they are using Traffic Information Service - Broadcast (TIS-B) from airports.

Stationary Traffic: The status of a runway can be set to active if there is moving or stationary traffic at the end of a runway. If an aircraft remains long enough in a restricted area the runway activity status is set to active. The assumption behind this is that an aircraft on this position is about to begin the take-off roll or not exiting the runway after a landing.

Stop Bar Violation Detection: Stop bar violations are often very dangerous situations, they happen whenever a vehicle crosses a hold line while entering an active runway from the taxiway. Stop bar violations are detected using different approaches. If tracking data is available the algorithm determines if the position of the vehicle moves across the stop bar. Due to the accuracy of the primary surveillance equipment this may lead to false alarms or missed alarms. False alerts can be minimized by using a margin around the stop bar that the vehicle must leave before an alert is raised. Short and medium range sensors like Universal Medium Range Radar (UMMR) or cameras, can be used to provide a reasonable surveillance accuracy at stop bars allowing the timely detection of stop bar violations. Alternatively some airports use inductive loops, microwave barriers or magnetic field sensors to directly detect stop bar violations. To determine the direction of traffic often at least two sensors are used, although some sensor provide vehicle signatures that can used to identify type and direction of travel of a vehicle. Both approaches may be combined, in such a configuration, usually the secondary surveillance system overrides the primary system, e.g. if a stop bar violation is detected by inductive loop sensors, while the information from the Surface Movement Radar (SMR) still indicates that the vehicle is likely still in front of the stop bar, an alert is issued based on the information from the inductive loop sensor. Unless the SMR detects the aircraft clearly behind the hold line, no alert will be issued if the inductive loop sensors give no indication that the aircraft is passing the hold line. Depending on the performance of the surveillance airports require pilots to stop their aircraft at least some meters before stop bars to avoid false alerts.

Stop Bar Violation Prediction: It is also possible to predict the violation of a stop bar if the position and speed of a vehicle can be estimated with sufficient accuracy. If a vehicle is approaching a hold line to an active runway or a closed (virtual) stop bar with such high speed that it cannot stop in front of the hold line the violation of the stop bar can be predicted before the violation happens. Often dedicated speed test areas are used

for this purpose. Note that the accuracy of primary surveillance equipment is often not as good as desirable for a reliable and timely detection if the speed of the vehicle is not very fast. Secondary surveillance equipment can be used to counter such problems.

Route Deviation Detection: To reliable detect route deviations an algorithm needs at least to know the routes assigned to each vehicle and the surveillance equipment must be able to identify the observed vehicles. If a vehicle is too far from its route a route deviation is detected. Route deviations are more difficult with non-cooperative vehicles where no id is available, especially if more than one of these vehicles are concerned. While it is still possible to define any taxiway and runway part that is not on the route as restricted area and detect a route deviation if a vehicle moves there, a vehicle may still move on the route of another vehicle. If routing information given via Radio telephony (RTF) is not conform to the route entered into the RIPAS nuisance alerts may result. A solution is the digital transmission of the route to the aircraft, but changes in the route via RTF can still occur. Although nuisance routing alerts are correct from a technical point of view, they disturb the work of ATCOs and may cause them to turn off the system.

Conflicting Clearance Detection: A RIPAS can detect or prohibit conflicting clearances e.g. it can refuse to open a stop bar leading to an active runway or prevent the different ATCOs from giving different aircraft clearance for take-off on intersecting runways. Of course ATCOs may still override RIPAS decisions via RTF.

Runway Incursion Detection Algorithms:

A range of algorithms have been developed in the past, and some are currently under development. Among these are:

AMASS: This early algorithm is not recommended for use by some studies [57]. Other sources found that AMASS generates alerts much slower than RIAAS or RIM [9]. AMASS has been found to be unreliable under certain conditions [102], and too slow to prevent runway collisions [80]. Given the constraints applied by surveillance and computational equipment available at the time AMASS was developed, the algorithm could not be expected to perform much better.

ASDE-X Safety Logic: The safety logic of the ASDE-X system is a set of algorithms that uses ASDE-X surveillance data as input to predict and detect conflicting traffic situations. The algorithms can also be used to control automatic warning and alert signals at the airport. ASDE-X has issued timely alarms in some runway incursion incidents.

RIAAS: The Runway Incursion Advisory and Alerting System (RIAAS) algorithm is designed for cockpit based operations to detect runway incursions involving the own aircraft. It generates two types of alerts, analogous to the Traffic Alert and Collision Avoidance System (TACAS) approach. Runway Traffic Alerts (RTA) should caution the flight crew of a potential dangerous situation while runway conflict alerts (RCA) require the pilots to take immediate action to avoid a collision. The National Aeronautics and Space Administration (NASA) published a description of the system in 2005 [9].

RSM: A description of the Runway Safety Monitor (RSM) algorithm used for the detection of runway incursions was published in January 2002 by the NASA [59]. Interestingly RSM relies strongly on a preprocessing of the data of the various sources and the first version could only handle 1 non cooperative traffic participant in a runway incursion area.

PathProx: PathProx is an aircraft based incursion alerting algorithm designed to handle over 40 different runway incursion scenarios [11]. It is the commercial Version of the RIAAS developed by the NASA in a commercial cooperation. A description of PathProx / RIAAS was published in May 2005 by the NASA [9]. The applicability of the algorithm in scenarios without sensor and transmission errors has been shown in a Monte-Carlo simulation.

RIM: The Runway Incursion Monitor (RIM) denotes algorithms used by the Deutsche Flugsicherung (DFS). After an incident in 2004 where the RIM had been disabled by ATC because of its unreliability, the German Federal Bureau of Aircraft Accidents Investigation (BFU) recommended that the DFS should improve the RIM to give reliable indications. This shows that European aviation administration had similar problems as the FAA with early RIPAS.

B-RSN: Basic Runway Safety Net (B-RSN) is a term for technology detecting runway conflicts based on A-SMGCS Level 1 and Level 2 information. PRIMA is a B-RSN class algorithm evaluated during the EMMA2 Project. Except for a few simulations runs where the traffic surveillance created phantom tracks resulting in nuisance alerts, the algorithm worked as expected. Further information on B-RSN and PRIMA can be found in an A-SMGCS Test Report [12].

A-RSN: An Advanced Runway Safety Net (A-RSN) is an improvement of B-RSN making use of A-SMGCS Level-3 services like electronic flight strips (EFS). Using this information route deviation detection is possible allowing an early resolution of a possible conflict. It can be considered as an add-on to B-RSN systems detecting aircraft in Runway Protection Areas (RPA) that do not behave compliant to the instructions given by ATC [12, 53, 32]. It has also a protection against operational errors that prevents the input of conflicting clearances. Due to establish best practices implementation of A-RSNs should be minimal with respect to the necessary rule set [32]. FISSA is an A-RSN class algorithm evaluated during the EMMA2 Project. Further information on A-RSN and FISSA can be found in an A-SMGCS Test Report [12].

XL-RIAS-Algorithm: Runway-Incursion-Prediction-Detection-Algorithm: This algorithm uses runway incursions zones similar to the RSM algorithm. The algorithm uses information from ATC as well as traffic surveillance data from local sensors to determine the activity status of a runway. The algorithm also performs an analysis of the speed and distance to the hold line to predict stop bar violations.

5.3.2 RIPAS systems

In the past, a number of solutions for runway incursion avoidance and detection were proposed, including the Runway Incursion Prevention System (RIPS), the Airport Surface Detection Equipment - Model X (ASDE-X), the Runway Status Light System (RWSL), the Airport Movement Safety System (AMASS), the Final Approach Runway Occupancy System (FAROS), The Runway Incursion Monitoring, Detection, and Alerting System (RIMDAS) and the Advanced Surface Movement Guidance and Control System (A-SMGCS). Of these, only A-SMGCS and ASDE-X in conjunction with RWSL and FAROS have been deployed at a significant number of major airports.

- RIMCAS: The Runway Incursion Monitoring and Conflict Alert System (RIMCAS) uses vector calculations to predict possible conflicts between aircraft or other objects
- RAAS: The Runway Awareness and Advisory System (RAAS) was developed by Honeywell.
- RIMDAS: The Runway Incursion Monitoring, Detection, and Alerting System is a design for an inexpensive local surveillance system that provides audible alerts to involved flight crews. RIMDAS is not comparable to existing technology solutions because it is based on some strong assumptions about the performance of experimental sensors and data fusion algorithms.
- AMASS is a system that was originally designed by the FAA for runway incursion detection. It failed its expectations because of bad prediction and a high frequency of false alarms. The focus has shifted to runway collision prevention [102], but 2005 the National Transportation Safety Board (NTSB) found that AMASS does not provide alarms early enough to prevent runway collisions .
- RWSL provides a means of protection by automatically setting the lighting of the runway in such a way that flight crews will notice whether a runway is occupied or not. Depending on the sensors that are used, the system could be too slow to react immediately and will thus prevent only some of the runway incursions. According to the specifications of the FAA, the RWSL uses traffic data from the ASDE-X as primary input [3]. RWSL is currently deployed to 23 U.S. airports. More Information can be found online at [65].
- FAROS is a system based on light cues to warn flight crews and ATCs during a final approach of a runway if the runway is or becomes occupied. FAROS require input from a surveillance system to determine the status of a runway. Experimental setups have been using inductive loop sensors embedded in the runway. The Version at DFW uses the ASDE-X system to monitor the entire runway surface.
- ASDE-X is a Fusion of SMR, Multilateration Radar and ADS-B that provides enhanced HMI to ATC. This system achieves a better resolution in time and space as an SMR-based system as well as a better accuracy of position reports. However, depending on the airport topography, a complete coverage of the airport surface is seldom possible, and problems have been found with non-cooperative aircraft and ground vehicles. ASDE-X Safety Logic (AXS) is expected to enhance the situational

awareness of ATCs by detecting possible collisions early and provides visual and audible alerts to controllers. ASDE-X installations are expensive, and its surveillance of hot spots is not as reliable and accurate as systems using localized sensors could be.

- RIPS was developed at NASA in 2001. This system relies on algorithms that are applied to the traffic situation, as observed by the traffic surveillance of airports. Thus, its performance depends heavily on the performance of the tracking and surveillance equipment that is available. For most airports, these instruments that provide data for immediate reactions consist of long range sensors with a limited resolution in time and space. Additionally, the system provides input via a heads up display that requires a retrofit of most cockpits, making it extremely expensive.

- PathProx is a system that is developed from RIPS. It provides runway incursion warnings and runway incursion alerts to flight crews who are equipped with PathProx. PathProx relies on ADS-B and TIS-B as input to its algorithms.

- A-SMGCS is an improvement of the SMGCS. Although studies on the Surface Movement and Guidance Control System included localized sensors already in 1998 [75], the current installations of A-SMGCS require only SMR, Multilateration and ADS-B for operations but can also integrate other sensors. A-SMGCS installations are expensive, and the surveillance of hot spots is not as reliable and accurate as systems using localized sensors could be. A-SMGCS has four levels of operation: surveillance, monitoring, planning, and guidance and control.

- XL-RIAS is an experimental localized runway incursion alerting system that focuses on the detection of planes entering a runway from a taxiway. This system can be used as a standalone solution, to enhance the sensor accuracy at hot spots, and to provide sensor coverage in blind spots.

- RunwayGuard is based on the integration of micro magnetic field sensors into ASMGCS; using only passive sensors, it has interesting capabilities compared to induction loops and other localized sensors.

- Mobile Application Based Systems (MABS) are software solutions for hand held devices with built-in GPSs. However, the accuracy of the dynamic position estimation of such devices is often not as good as required, and there are open issues regarding the safety and security of this approach. Still, this approach could be an interesting solution for GAs.

5.4 Performance Simulation Study

This section addresses the performance of the available runway incursion systems. It is strongly based on the results of several studies on the performance of Runway Incursion Prevention and Alerting Systems (RIPAS) that have been published [9, 119, 104, 10, 109, 53, 59, 23, 110, 12, 55]. Because of the different setups of the incursion scenarios, these are seldom directly comparable. Data on the performance of Airport Surface Detection Equipment- Model X (ASDE-X), Runway Status Lights (RWSL) and Final Approach

Runway Occupancy Signal (FAROS) is published in the Federal Aviation Administration (FAA) Annual Safety Report. The Advanced Surface Movement Guidance and Control System (A-SMGCS) concept has been tested in comparative setups during European Airport Movement Management by A-SMGCS (EMMA) and European Airport Movement Management by A-SMGCS 2 (EMMA2) projects. The National Aeronautics and Space Administration (NASA) has published studies on Runway Incursion Prevention System (RIPS), RIAAS and RMS. These studies assume the availability of Automatic Dependency Surveillance-Broadcast (ADS-B) and data links, so incursions with non-cooperative traffic participants are hardly covered. In some cases, independent studies on surveillance technology [77, 78] have been used to estimate performance parameters where no detailed information could be found in the RIPAS studies. Missing data have also been supplemented by the simulation of a runway incursion scenario with different sensor setups that are typically used for ASDE-X and A-SMGCS, for cooperative and non-cooperative traffic participants. To evaluate the performance of a runway incursion system, it is useful to begin with the causes of runway incursions. As mentioned in the previous sections, Pilot Deviations (PD) and Operational Errors (OE) contribute the most to class A and B Runway Incursions. Therefore, the focus is on the measures of protection against faults leading to runway incursion. It is useful to distinguish between the measures of protection for the prevention of runway incursions and the performance when addressing runway incursions that happen despite the operation of warning systems, i.e., the alerting performance. These functionality depends on the surveillance function of the system. In the study from NASA, all of the false alerts and missed detections could be traced back to sensor and data transmission errors [119]. Similar findings during the EMMA2 project [110] and other studies on A-SMGCS [62] support the position that the reliability of the surveillance is of utmost importance. Apparently, the maturity of the surveillance technology and the local circumstances set limits for the operational performance of the runway incursion prevention technology.

5.4.1 Surveillance Performance

For ASDE-X and A-SMGCS, requirements have been specified by the FAA and the International Civil Aviation Organization (ICAO). With respect to accuracy, these requirements determine the minimal performance of the system for determining the position, orientation, speed and identification of the vehicles at the airport. Similar requirements can be assumed for cockpit-based systems. Both the ASDE-X and A-SMGSCS requirements are meant as a lower bound for the surveillance performance; a better performance is recommended. In the EMMA2 project, it was found that no technology currently fielded can satisfy some of the requirements for A-SMGCS systems and that the specifications should be reconsidered in some points.
Nevertheless, technical solutions for both systems are available for cockpit based, A-SMGCS and ASDE-X systems. For both systems, the surveillance of cooperative mobile units is, in general, much better than the surveillance of non-cooperative mobile units. It will, therefore be distinguished between the surveillance of cooperative and non-cooperative mobile units.

Indicator	ADS-B	SMR(i)	MLAT
Req.position accuracy	Sigma4 m	Sigma+bias 5.9 m/6.5 m	Sigma+bias 6.5 m/7.5 m
Emp. position accuracy	Often>4 m	Sigma+bias 12-15 m	Sigma+bias 7.4 m/11 m
Req. probability of detection	0.995	0.9	0.92/0.999
Emp. Probability of detection	0.995	0.9	1.0/0.998
Req. Probability of false detection	n.a.	0.00001	0.02
Emp. Probability of false detection	n.a.	0.00001	0.02
Req. Target report update rate	1 Hz	1 Hz	2 Hz
Emp. Target report update rate	1 Hz / 0.1 Hz	1 Hz	2 Hz
Req. Delay	0.25 s	0.25 s	0.25 s
Emp. Delay	0.25 s / 2.0 s	0.25 s	0.25 s / 0.5 s

Table 5.3: Airport Surface Movement Performance Requirements for Sensors. Sigma is calculated from the Estimated Position Uncertainty (EPU) or the Horizontal Figure of Merit (HFOM), assuming a circular uncertainty ellipse. Lines beginning with req. show the requirements for (ASDEX/ASMGCS). Lines beginning with emp. show the performance experienced in field studies (ideal experiment/ working conditions experiment) [109, 77, 78, 2].

Performance Requirements

For ASDE-X and A-SMGCS, the performance requirements for the surveillance system are quite similar. ASDE-X specs also provide requirements for the sensors systems used; for A-SMGCS, the sensors are required to perform at least to the general surveillance requirements; there are also Minimal Operational Performance Specifications (MOPS) for the sensors that are available [29]. Dedicated functionality of cockpit-based systems, e.g., on-board guidance, have much higher requirements than airport-based systems. These requirements are assumed to be sufficient to detect and prevent most runway incursions. Because the required accuracy/uncertainty is often given in different measures for different sensor subsystems, the accuracy/uncertainty in the following text is given by sigma deviation assuming a bivariate normal distribution of the position estimates, as calculated from the requirements. Table 3 provides an overview on the sensor requirements for the ASDE-X and A-SMGCS systems. However, the dedicated functions of A-SMGCS or ASDE-X may have deviating requirements, e.g., A-SMGCS specs assume 20 m accuracy as sufficient to detect runway incursions in time. This result should, however, not be considered as sigma 20 m but, rather, should be considered to be reliable information for reducing false alarms and detecting runway incursions in a timely fashion. At runway incursion rates of 16 (FAA) to 60 (European Organisation for the Safety of Air Navigation (EUROCONTROL)) RIs per million operations, even an accuracy of sigma 5 m would lead to a high number of false alerts, given the calculation basis in the A-SMGCS Manual.

Surveillance of Hot Spots

Given the definition of hot spots, the surveillance of hot spots should satisfy higher requirements than general surveillance. In hot spots, movements on the taxiway and the runway could lead to a conflict in a short time, where general surveillance requirements are not sufficient to detect conflicting situations as quickly as desired. While the ASDE-X and A-SMGCS do not explicitly require a better surveillance of hot spots, bigger airports employ localized surveillance technology in hot spots, particularly for the surveillance of

hold lines.

Surveillance of Cooperative Mobiles

For A-SMGCS and ASDE-X equipment, the surveillance of cooperative mobiles, which include humans and vehicles equipped with active and/or passive transponder technology, is more reliable and accurate. The reason is that, in addition to the data from the Surface Movement Radar (SMR), the data from the cooperative sensor systems can be used to acquire information that is competitive and complementary to the data from the SMR, resulting in an increase in timeliness, certainty, accuracy and completeness of the surveillance data. A remaining problem is the position of the Multilateration Mode S Radar (MLAT) antennas on the aircraft. The antennas are usually not positioned at the center or nose of the aircraft, resulting in a systematic error of the estimated position of the vehicle. While this error could be negligible for vehicles such as cars, it can be a problem for larger aircraft. In addition, antennas are frequently not working or turned off. With ADS-B, other problems exist. The first problem is that most aircraft equipment that provides the information transmitted via ADS-B does not comply with the requirements of A-SMGCS or ASDE-X specifications with respect to the accuracy of the data. The second problem is the loss of data during transmission. Nevertheless, surveillance accuracy of MLAT, ADS-B and Surface Movement Radar Improved (SMRi) systems has been shown to satisfy A-SMGCS and ASDE-X requirements under ideal circumstances, e.g., with a special test vehicle at midnight on an otherwise empty airport. Under working conditions, it was found that the requirements were often not satisfied, especially under disadvantageous circumstances. Table 5.3 summarizes some of the findings of the field studies.

Surveillance of Non-Cooperative Mobiles

For the surveillance of non-cooperative mobiles, data from the SMR is often not reliable, timely or accurate enough for the implantation of the runway incursion technology that Air Traffic Controllers (ATCO) rely on. The insufficient accuracy of such systems leads to a high number of false alerts and missed alerts. Consequently, the systems have been turned off by ATCOs. The required probability of detection pd for a SMRi is 0.9, thus the probability of missing the same target for 2 seconds or longer is 0.019. For a reliable detection of runway incursions by non-cooperative targets, local surveillance sensors must provide the necessary accuracy and reliability when dealing with non-cooperative mobiles. Recent advances in the development of automotive radar sensors made inexpensive sensors well-suited to such tasks widely available. Systems based on thermal and high resolution cameras that are well-suited to monitor runway entrance areas and to hold lines are also available.
For the FAA, the surveillance of non-cooperative targets could become less pressing in the future because all aircraft in the USA will be required to equip ADS-B transponders by 2020, however the FAA is also researching inexpensive airport surveillance systems for smaller airports [83].

Simulation Study

A simulation study of the surveillance performance was done to provide an independent comparison of the different sensor technologies that are used by RIPAS. Data fusion is performed with a particle filter, to estimate the position, heading and speed of the traffic. The study examines 2 vehicle movements at a taxiway / runway intersection with 8 different sensor setups. The setup of the scenario is shown in Figure 5.4.1. The location of the setup is representative for this class of runway incursions, the pilots may not be able to see the approach and the runway threshold as they are facing the opposite end of the runway during the approach and the hold line is close to the runway. The circumstances in the simulated setting are similar to those encountered in the two runway incursion examples in Section 5.1 and many other category A and B runway incursions.

Figure 5.14: The setup of the simulation experiment. Left: Schematic map: The small red triangle marks the position of the vehicle that follows the green marked way points. The two green polygons indicate the boundaries of the pre hold line area and the post hold line area that are used to determine the distance of the vehicle to the hold line. The hold line is located at the intersection of the two areas. The reddish arcs mark the fields of view of the two local sensors. Right: The setup as perceived in the simulator. The aircraft is approaching the hold line with the illuminated stop bar.

In the first traffic scenario, a vehicle approaches the hold line from the taxiway, decelerates (app. 1.2 m/s^2) approximately 35 m before the hold line until it is approximately 7 m from the hold line, and comes to a complete stop approximately 3 m in front of the hold line. It stays there for approximately 30 seconds; then accelerates (app. 2,5 m/s^2) to a speed of app. 40 km/h and continues its way on the runway. In the second scenario, a vehicle approaches the hold line from the taxiway, crosses the hold line and continues its way on the runway without stopping or significantly slowing down. There are no phases of high acceleration or deceleration in the immediate vicinity of the hold line .
These two scenarios are typical examples of the interfering traffic movement encountered during a runway incursion from the taxiway and the non-inferring traffic encountered during normal operations. The setup of the sensors and the vehicle movement provide a good starting point to analyze the information basis for RIPAS alert decisions and situation assessments.
The sensor setup for both scenarios consist of the combination of MLAT, ADS-B and SMR, as favored by A-SMGCS and ASDE-X (Baseline Surveillance (BSurv)), and the same combination extended by UMRR sensors (Extended Surveillance (ESurv)). The

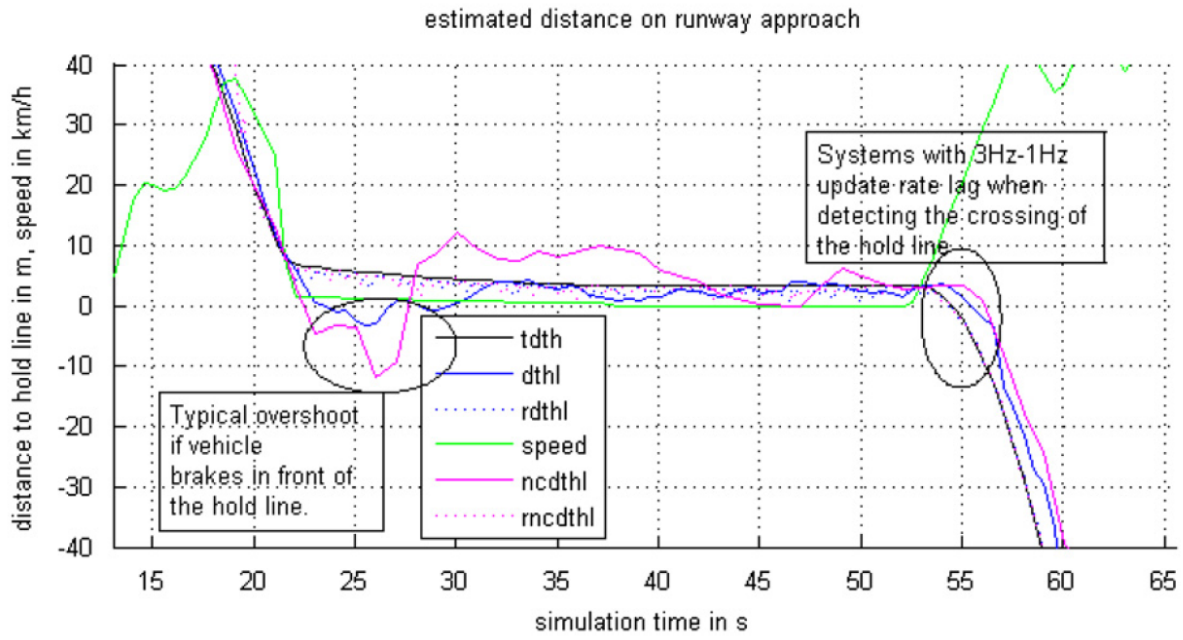


Figure 5.15: A representative example of surveillance performance results from simulation for traffic scenario 4, with sensor performances according to requirements specifications. The line dthl plots the true distance of the vehicle to the hold line and the line dthl plots the estimated distance to the hold line (EDTHL) with Baseline Surveillance (BSurv) and a cooperative target. The line ncdthl plots EDTHL for BSurv and a non-cooperative target. Typical for these approaches are the overshoot when the vehicle brakes and the lag when the vehicle accelerates again. The line rdthl plots the EDTHL for Extended Surveillance (ESurv) and a cooperative target and the line rncdthl plots the EDTHL for ESurv and a non-cooperative target.

first two sensor setups simulate BSurv and ESurv with a cooperative target, and the next two simulate Bsurv and ESurv with a non-cooperative target. For these four settings, the sensor performance satisfies the requirements of ADSE-X and A-SMGCS. The last four setups are the same as the first four, but the sensor performance has been adjusted to reflect the findings of field studies regarding sensor performance. Figure 5.15 shows a typical outcome for the first 4 sensor setups and the first traffic scenario.
The primary finding of the simulation study was that, in the second traffic scenario, all of the configurations follow the movement of the vehicle quite well. However, in the first traffic scenarios, BSurv configurations usually follow the vehicle during deceleration and acceleration not as well as during continuous movements. This scenario results in an overshoot when the vehicle stops at the time window 20-30 seconds (see Figure 5.15). This overshoot makes it difficult to distinguish between a vehicle stopping directly in front of the hold line and one that crosses the hold line for a few seconds immediately after the vehicle stops or crosses the hold line. The BSurv configuration also lags behind the position of the vehicle once the vehicle begins to accelerate and crosses the hold line at the time window of 50 to 60 seconds (see Figure 5.15). During the hold of the vehicle, the estimated distance to the hold line fluctuates a few meters around the true distance to the hold line. These findings apply specifically to the non-cooperative target

setup. The ESurv configuration, in contrast, has no overshoot and immediately follows the accelerating vehicle. The consequences for the alerting performance are discussed in Section 5.4.3.

5.4.2 Prevention Performance

The primary task of a RIPAS is to work better than humans, who rely on technology that is available at a specific time. Therefore, it appears reasonable to measure the performance of RIPAS on the decrease of runway incursion severity and frequency. Based on the statistics of the FAA ASDE-X, RWSL and FAROS installations at Dallas/Fort Worth International Airport (DFW) reduced the number of runway incursions by approximately 70%, showing that even though the technology does not satisfy the requirements under all of the circumstances, the increase in safety is measurable. Notably, these systems work by increasing the situational awareness of the traffic participants on the runway and taxiway by providing information about the runway activity status and the status of hold lines. A EUROCONTROL study found that the use of stop bars for 24 hours resulted in an increase in safety [86]. Prevention performance can be further increased by integration of Electronic flight Strips (EFS), routing and clearance information, as EUROCONTROL studies show [5]. A major current problem appears to be with the ATC - HMI. Experimental Human Machine Interfaces (HMI) have been experienced as difficult by ATCOs, especially concerning automated routing functions; also, under some working conditions, ATCOS prefer the view out of the window and need haptic feedback interfaces because they do not want to look down on the airport map display to change a stop bar state or to perform a similar operation [86].
Cockpit-based systems have been shown to prevent runway incursions by detecting route deviations and by warning the pilot of converging traffic. They also show the locations of hold lines and other operational topographic points relative to the aircraft. This enhanced situational awareness has its maximum effect under conditions of limited visibility. The performance degrades with non-cooperative traffic because this traffic is accessed through Traffic Information Service-Broadcast (TIS-B) and with less accuracy. Still, Electronic Flight Bag (EFB) can afford to provide warnings in such situations without creating a nuisance to the pilots because take-off and landing operations usually make up for only a small part of the overall flight time.

5.4.3 Alerting Performance

This section covers the performance of RIPAS in case there is an incident in which a runway incursion has already happened. The main goal of the RIPAS is to predict or detect the incident as fast as possible and to provide an immediate alert to the involved parties. Because evaluation studies are not totally comparable, a simulation was conducted to estimate some of the performance parameters for a selected runway incursion scenario.

Performance estimated from Evaluation Studies

Quantitative performance measures frequently encountered in evaluation studies are the increase in the separation distance, the number of missed alerts, and the number of false and nuisance alerts. In some studies, actual runway incursion incidents or accidents are

analyzed in simulations and field tests, with no RIPAS and different RIPAS configurations. The findings of these studies show that the maximum effect is given for cooperative targets and low visibility conditions. It was found that, for some runway incursion incidents, it was not possible to avoid an accident without the use of a RIPAS because of the insufficient situational awareness of the pilots, even under good visibility conditions. The separation distance was increased by several hundred feet for many runway incursion scenarios. An overrun of hold lines could be shortened by airport-based systems and could be reduced even further by cockpit-based systems. However, most of the simulation studies assumed a surveillance that outperforms the requirements, and field tests were often performed with very good equipment. Performance evaluations under real operating conditions revealed that the equipment found on aircraft or vehicles is not comparable to the equipment used in the field test and that the performance of the surveillance does not necessarily meet the specified requirements. The results of such system evaluations are, therefore, an example for ideal conditions, assuming better surveillance technology in the future.

Simulation Performance

RIPAS evaluation studies are not directly comparable because of different test setups, environmental conditions and other factors. Thus the alerting performance was estimated by simulations, which allows for a comparison of different test setups. To characterize the alerting performance, the following evaluation criteria are introduced:

1. time to detect conflict t_{dc}: The time that passed from when the runway conflict occurred until the system reliably detects the incident. Generally, an increase in t_{dc} can make up for a lack of accuracy in the surveillance with respect to the probability of missing an alert pma due to insufficient space for a safe separation of vehicles.

2. total time to alert t_{tta}: The time until the alert has arrived at the involved parties.

3. probability of missed alert p_{ma}: The probability of missing a runway incursion alert strongly depends on the airport topography, especially the distances between hold lines on runways and the accuracy of the surveillance equipment.

4. probability of false alert p_{fa}: The probability of creating an alert without a conflicting situation. This probability strongly depends on the airport topography, especially the distances between hold lines and runways and the accuracy of the surveillance equipment.

For most systems, the performance in each of these criteria depends strongly on the surveillance/tracking performance of the system and is different for the various runway incursion scenarios that are indicated by many authors [9, 10, 59]. Therefore, the alerting performance of the systems is determined based on the results of the simulation in section 5.1.
The primary finding was that, with sensors performing in accordance with the requirements, missed alerts did not occur but false alerts and late alerts could occur. Missed alerts occurred and the number of late and false alerts increased for simulation Scenario 1 with sensor performance as gained during field studies, where the sensor performance was

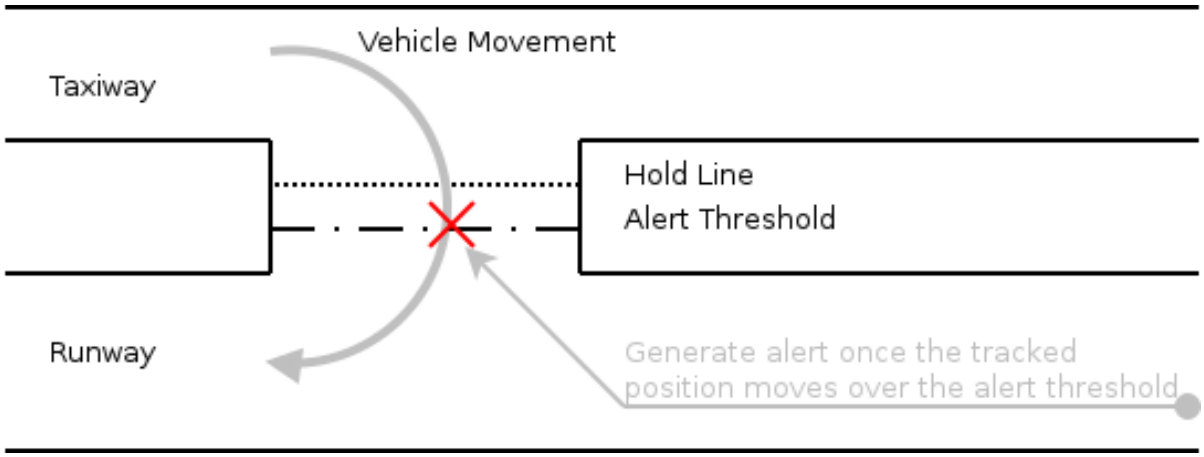


Figure 5.16: Distance to hold line based algorithm. The algorithm generates an alert when the tracked position of a vehicle moves over a threshold value beyond the hold line.

not as good as required. This result is caused by the degradation in the surveillance performance, resulting in an increase of the overshoot and the lag during tracking, increasing the effects shown in Figure 5.15.
To determine the results, a distance to the hold line-based algorithm was applied to the surveillance results from the simulations as well as from different setups for the notification of the involved parties. This algorithm raises an alarm if a vehicle travels beyond a predefined distance over a stop bar. The primary findings are that the time to detect a runway incursion t_{dc} and the probability for a false alert p_{fa} were closely correlated; thus, at the expense of t_{dc}, p_{fa} can be decreased to an acceptable margin. This result can be achieved by raising the alert threshold distance to a distance to the hold line that is based on an algorithm or requiring the target to persist at least for some time at a position beyond the hold line. Figure 5.21 shows this relationship. Before discussing this finding in detail the results from the two traffic scenarios will be discussed and compared two each other.

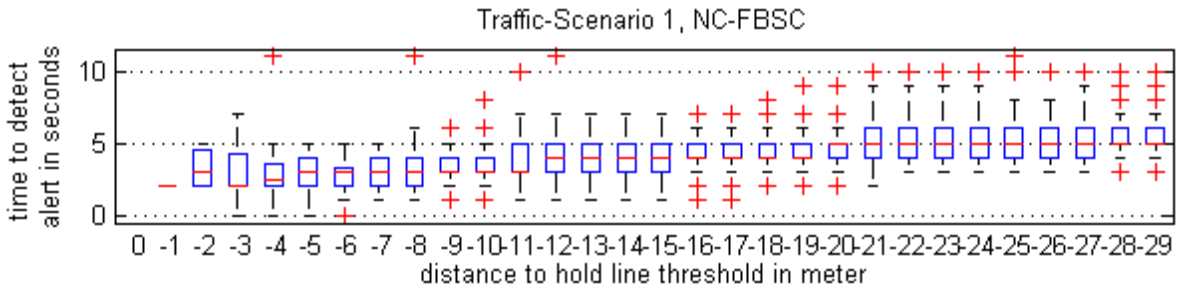


Figure 5.17: Statistical evaluation of the influence of the alert threshold, based on the distance to the hold line. An alert is issued once the vehicle's distance from the hold line exceeds the threshold. The chart show t_{dc} for the first traffic scenario, with a non-cooperative target and SMR only tracking for the surveillance of the hold line.

Scenario 1: For both scenarios the setup of the simulated environment is the same, see Figure 5.4.1 for details on the setup. The first scenario represents a situation every RIPAS should be able to cope with . A vehicle approaches a hold line and holds before the hold line awaiting clearance to proceed onto the runway. It stays there for app. 30 seconds, then continues on the runway. There are two challenges for the RIPAS: The first challenge is that no false alarm should be raised while the vehicle is waiting. The inability of many previously and currently deployed RIPAS to do so has caused ATCOs to turn of the RIPAS notification. The decision of the ATCOs is quite sensible as a false alarm would direct the attention of the ATCOs to the wrong places and evasive maneuvers that may be triggered by the false alarm are often risky. The second challenge is to detect a runway incursion as soon as possible, once the vehicle is moving beyond the hold line. According to representatives of the NTSB, a total time to alert of more than a few seconds it to much []. Figure 5.17 shows the statistical distribution of the alerting times for distance to hold line thresholds (DTH) between 0 and 29 meters. Notable is that, although the vehicle stops approximately 4-5 meters in front of the hold line, the standard sensor setup, top chart in the figure, raises a false alert at DTH 0m and -1m in almost every simulation run. The reason is the overshoot of the vehicle when it stops at the hold line, see Figure 5.15. With respect to the t_{dc} DTH -2m to DTH -19m is about 3-4 seconds, a stable trend is visible only below DTH -12m. The reason is the high number of false alerts at DTH between 0m and -11m, leaving not enough results for the statistical evaluation. Not shown here are the missed alerts, where the system took more than 30 seconds to detect the runway incursion.

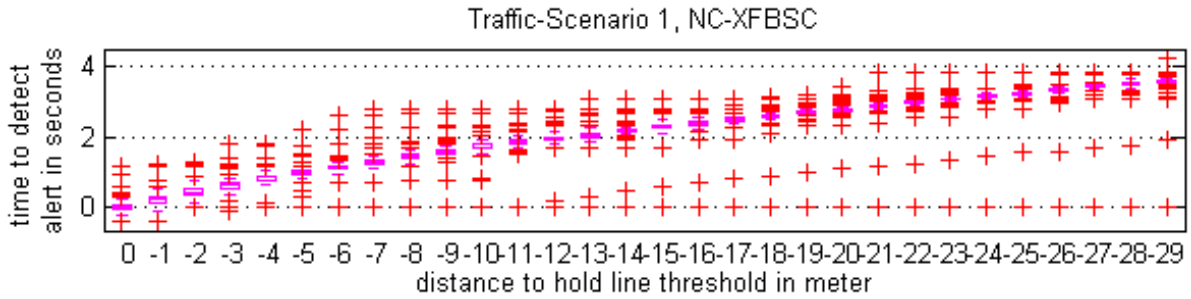


Figure 5.18: Statistical evaluation of the influence of the alert threshold, based on the distance to the hold line. An alert is issued once the vehicle's distance from the hold line exceeds the threshold. The chart shows t_{dc} for the first traffic scenario, with a non-cooperative target and additional use of localized sensors for the surveillance of the hold line.

When using localized sensors the alerting performance of the system increases significantly. Figure 5.18 shows the statistical distribution of the time to detect conflict t_{dc}. Notable is the visible trend through all DTHs and the low variance of detection times. There where no missed alerts encountered during the simulation runs. There is a number of outliers but none of these are than 2 seconds later than the mean detection time of the corresponding DTH. Compared to the standard sensor setup the detection time when using localized sensors is much lower, between 0 to 4 seconds, compared to 2 - 5 seconds without localized sensors.
Notable is the line of outliers that raises from DTH -11 to DTH -29 about two seconds

below the mean detection time. A reason for the outliers may be particle deprivation. A particle deprivation happens when too few particles are in the vicinity of the true system state. This has been observed visually in the HMI. The cause is the inaccurate tracking by the SMR before the vehicle enters the area under surveillance by the localized sensors. The very accurate feature observations provided by the localized sensors cannot be associated with the particles of the track, causing the system to abandon the track and reinstate a new track. The problem can be fixed using a higher number of particles, but this approach was too slow to run thousands of simulation runs.

Scenario 2: In the second scenario the vehicle follows the way points at steady speed without phases of significant acceleration or deceleration. The main challenge to the tracking is the rapid change of heading in the intersection areas before and after the vehicle moves across the hold line, see Figure 5.4.1 for details on the setup.

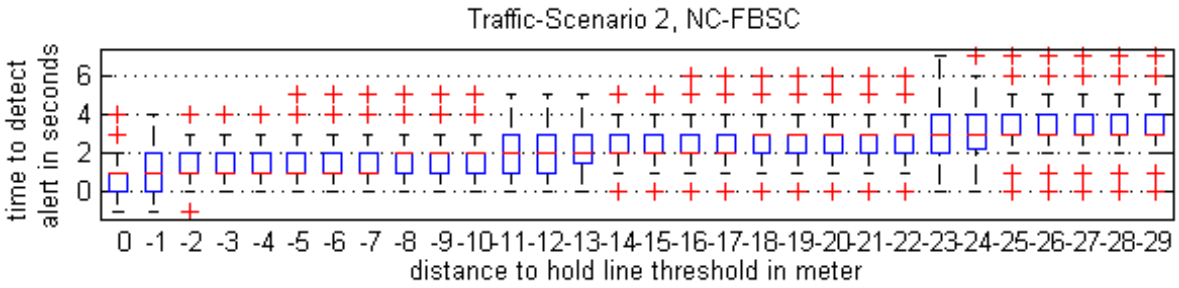


Figure 5.19: Statistical evaluation of the influence of the alert threshold, based on the distance to the hold line. An alert is issued once the vehicle's distance from the hold line exceeds the threshold. The two charts show t_{dc} for the second traffic scenario, with a non-cooperative target and SMR only tracking.

Figure 5.19 shows the detailed results for the standard sensor setup. Compared to the first traffic scenario the runway incursion detection performance is better. There is a visible trend through the whole DTH range because there are fewer false alerts and fewer missed alerts. The time needed to detect the runway incursion is also reduced, but due to the relative high speed, compared to the first traffic scenario, the proportional increase in separation distance after an alert is less than the proportional decrease of the detection time. Also the higher speed results in the vehicle crossing the DTH thresholds earlier.
The performance of the sensor setup using localized sensors is comparable to the performance in the first traffic scenario, the reduced alerting times are just caused by the fact that the vehicle has a higher speed and thus reaches the DTH threshold earlier. Noticeable is the fact that there are fewer outliers, an evidence that supports the observation that the inaccurate tracking before entering area under the surveillance of the localized sensor results in a particle deprivation.
The chart in Figure 5.21 shows that t_{dc}, p_{ma} and p_{fa} directly correlate with the threshold distance. Even with localized sensors, there is always a probability for a false alert, although no missed alerts occur. Thus, the localized sensor setup reaches a reasonable $p_{fa} = 0.02$ with a threshold of -10 meters; the ASDEX/ASMGCS sensor setup reaches the same p_{fa} with a threshold of -19 m. At this point, the value of t_{dc} is approximately

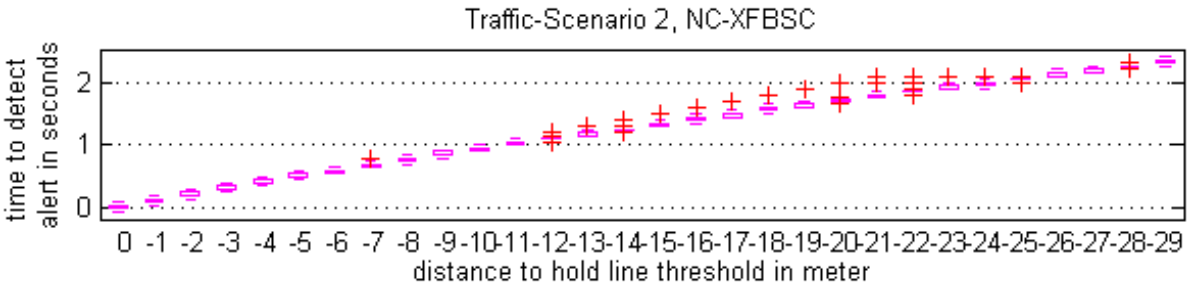


Figure 5.20: Statistical evaluation of the influence of the alert threshold, based on the distance to the hold line. An alert is issued once the vehicle's distance from the hold line exceeds the threshold. The chart shows t_{dc} for the first traffic scenario, with a non-cooperative target additional use of localized sensors for the surveillance of the hold line.

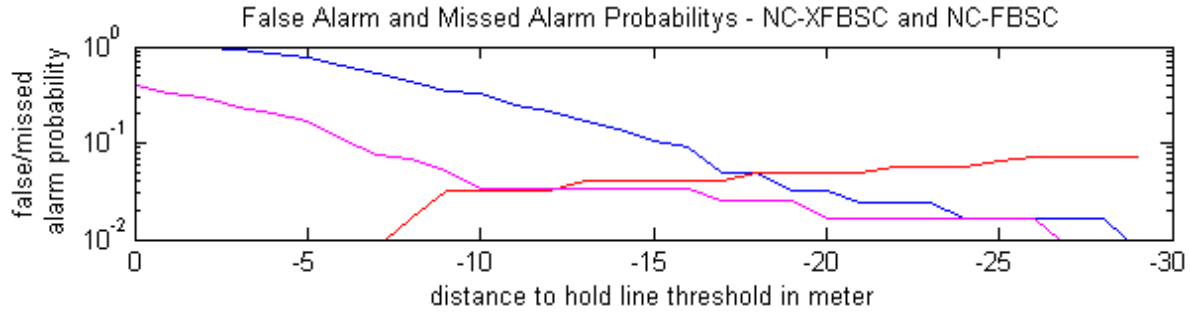


Figure 5.21: Statistical evaluation of the influence of the alert threshold, based on the distance to the hold line. An alert is issued once the vehicle's distance from the hold line exceeds the threshold. The graph shows p_{fa} and p_{ma} for these setups. The blue and magenta line are the p_{fa} for SMR, only tracking (blue) and localized sensors (magenta); the red line is the maximum missed alert probability for SMR only, tracking in both scenarios.

2 seconds with localized sensors, respectively, 4-5 seconds without localized sensors. The localized sensor thus has an advantage of 2-3 seconds in t_{dc} and a much lower p_{ma}.
However, an increase of t_{dc} directly increases t_{tta} and could limit the number of usable signals. For example, a non-cooperative target traveling with 15 m/s over a hold line may, with an insufficient update interval and an alert threshold of 20 m, have traveled 40 m before the detection of the alert situation. Delays from the signal command transmission and the reaction time of the signals require at least 0.5 s, usually more, until the signals at the runway entrance can be activated. Depending on the distance between the hold line and the runway, the pilot may be unable to see any alert signals. An alert notification via RTF to the pilot will require additional time because the ATCO needs to react to the alert and inform the pilots. Estimating 1 second ATCO reaction time, and 2 seconds for communication, the difference in t_{tta} between a localized sensor with localized signals and a standard ASDEX/ASMGCS setup without signals might be as much as 6 seconds on average. The values in Table 5.4 show that the distance covered in both scenarios until the pilots responds to the situation is for the first acceptable DTH 15m/36m for the localized sensor setup, an 84m/105m for the standard setup. The advantage in horizontal separation distance of the localized sensor is 63m/69m. The advantage is correlated to

sensors	signals	DTH	t_{dc}	t_{cd}	t_r	t_{total}	p_{fa}	p_{ma}	d_{hl}	valid
standard	voice	0m	-/1	-	-	-	1	0	-	no
extended	local	0m	0/0	0.5	1	1.5s/1.5s	0.5	0	1m/23m	no
standard	voice	-5m	-/2	-	-	-	0.9	0	-	no
extended	local	-5m	1/0.5	0.5	1	2.5s/2.0s	0.1	0	2m/30m	no
standard	voice	-10m	3.5/2	2	2	7.5s/6.0s	0.5	0.02	70m/90m	no
extended	local	-10m	1.8/0.9	0.5	1	3.3s/2.4s	0.02	0	15m/36m	yes
standard	voice	-15m	4/2.5	2	2	8s/6.5s	0.1	0.03	77m/98m	no
extended	local	-15m	2/1.3	0.5	1	3.5s/2.8s	0.02	0	20m/42m	yes
standard	voice	-20m	5/3	2	2	9s/7s	0.025	0.03	84m/105m	yes
extended	local	-20m	2.5/1.6	0.5	1	4s/2.9s	0.01	0	32m/44m	yes

Table 5.4: Alerting performance comparison, probabilities are given per operation. In columns t_{dc}, t_{total}, and d_{hl} the first number is for traffic scenario, 1 the second number for traffic scenario 2. t_{total} is the time in seconds until the pilots response. d_{hl} is the distance traveled across the hold line towards the runway center line.

the speed of the vehicle, 15m/s was chosen as a sensible measure, but speed has often been reported to be much higher during taxiing.

5.5 System Architecture - Safety Based Approach

The superordinate goal of the airport research project RollMops was to increase runway safety. Therefore a safety based approach is necessary. In the consequence safety is a major aspect of the system design and places hard constraints on the design, from the mathematical models and algorithms to the realization. While the available project resources did not allow the realization of a certified prototype, the design process and the formalisms have been extended to support a safety based approach where needed. This section discussed the systems design with respect to safety and reliability of the system, with varying system boundaries. While safety and reliability are different concepts they overlap at some points, and while the two concepts are not completely orthogonal it is often appropriate to treat them as if they where, see Figure 5.22.
The section starts with an introduction to engineering safe systems and introduces the systems setup at the airport and the embedding into the existing airport infrastructure. The safety and reliability of the system with system boundaries of the surveillance and control of a single hold line are then discussed. The results are then integrated to show the estimated impact on the runway safety for airports of different sizes.

5.5.1 Safety Driven Engineering

A critical part of safety driven engineering is the safety analysis that becomes the central part of the engineering process. Safety analysis are done in all stages of the engineering process, with different goals, scope and limits. Because of the capabilities and responsibilities of the TIS Research Group during the project the scope of the analysis is limited to scientific and technical aspects of the design and the design process, see Figure 5.23. The responsibility of the TIS Research Group was focused on the development of a Lo-

Figure 5.22: Two visions on safety and reliability. Left: Safety and reliability are overlapping but not identical. The reliability of a system, or parts of the system can affect the safety of the system. Right: A gun is an example for a reliable yet unsafe device. The airport traffic control systems as they are now are quite reliable but the safety could be enhanced if the RIAS is part of the airport traffic control system.

cal Intelligence Unit (LIU) that should integrate the sensor data from localized sensors and ATC to control traffic lights at hot spots. The project partner ADB-Siemens was responsible for the remaining aspects of the system. The goals, scope and limits of the safety analysis in the context of the development and design process are the topic of the following text.

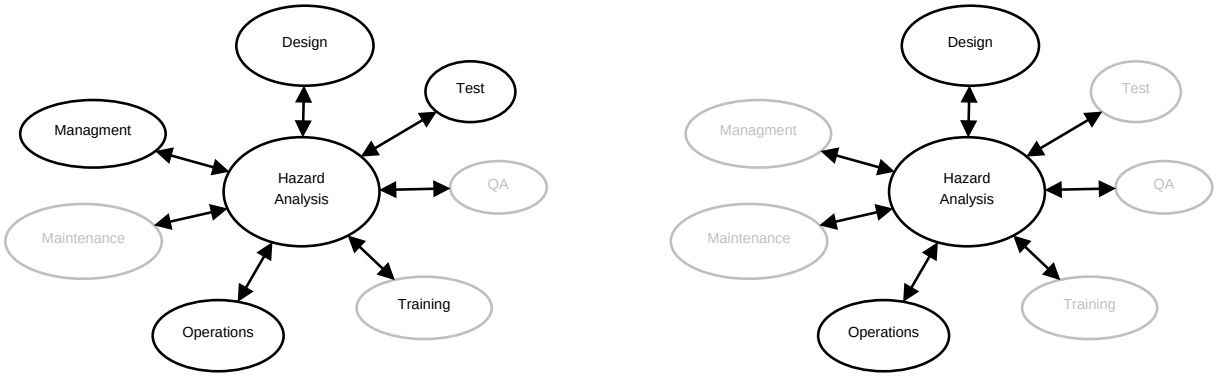


Figure 5.23: Scope of the safety analysis in the lufo project. Parts grayed out cannot be addressed. When considering only the Local Intelligence Unit (LIU) , to the left , that is responsible for the sensor data fusion and situation assessment more topics can be addressed than possible when considering the XL-RIAS, to the right, because the design and development of the LIU is, while subject to the constraints given by the specifications of ADB-Siemens, work of the TIS Research Group. The XL-RIAS includes factors that cannot be assessed by the TIS Research Group but are only accessible by ADB-Siemens.

Goal of the Analysis

The goal of the safety analysis depends on the stage of development. The stages are conceptual development, design, development, operations, and deployment. During the stages conceptual development, design and development the goal is the assessment and identification of possible hazards, and the elimination or control of these hazards. Also

the safety integrity level (SIL) is to be assessed, with respect to the hazards, risks an reliability of the system. In the operation stage the goal is to analyze data gathered during the operation of the deployed systems to identify problems that did not emerge during the previous stages of the process and put things right.

Scope of the Safety Analysis:

The scope of the analysis depends on the application scenario that sets the system boundaries to be considered during the analysis. With the goal to identify hazards the analysis usually starts with the system itself and its intrinsic potential for possible hazardous behaviors, e.g.: A industrial plant might release toxins into the air. The next step is to increase the boundaries to include the immediate environment of the system, like the neighborhood of an industrial plant and the relations between the immediate environment and the system, as well as the interactions between humans and the system. Finally hazards that pose a threat to the remote environment of the system must be identified, e.g.: A failure in a power plant might result in traffic accidents.

Limits of the Safety Analysis:

As for all logical or mathematical analysis the safety analysis is prone to errors induced by the gap between the real world and the formal and mathematical models that can only approximate the real world. The analysis is by no means a verification that guarantees that the systems behavior will be in the scope determined by the analysis. The reasons for this are that many requirements related to the personnel and tools used during the design process, as well as demands on the design process cannot be satisfied in the context of an academic research project. The analysis rather shows how existing formalisms and methods can be extended to support the engineering of safety systems, and estimates the reliability and the impact on the airport operation safety.

Approach

Since there is no absolutely safe system, approaches design safe systems aim to build systems that are reasonable safe with respect to a given application scenario. The safety of a system can be measured by its SIL that determines the risk a system poses in its application scenario. The classification into a SIL is done based on the reliability of the safety related functionality of the system, and the corresponding risk to the environment of the system. The hazard analysis is considered the central point of a safety analysis by Leveson [67] and Douglas [27] that distinguish between:

- Preliminary Hazard Analysis (PHA)
  PHA is used in the early life cycle stages to identify critical system functions and broad system hazards. This term introduced by Leveson is roughly equivalent to the term “Initial Safety and Reliability Analysis” used by Douglas.

- System Hazard Analysis (SHA):
  SHA begins as the design matures and involves detailed studies of possible hazards created in the interfaces between subsystems or by the system operating as a whole, including potential errors.

- Subsystem Hazard Analysis (SSHA):
  SSHA starts as soon as subsystems are designed in sufficient detail. The purpose of SSHA is to identify hazards associated with the design of the subsystems. Software Hazard Analysis is a type of SSHA.
- Operating and Support Hazard Analysis (OSHA):
  OSHA identifies hazards and risk reduction procedures during all phases of system use and maintenance.

Douglas summarizes the most relevant steps in a hazard analysis as:

1. Identify the hazards
2. Quantify the hazards in terms of likelihood and severity
3. Compute the risks(likelihood x severity)
4. Perform an initial safety analysis with Fault Tree Analysis (FTA)
5. Perform an initial reliability analysis with Failure Mode and Effect Analysis (FMEA)
6. Identify safety and reliability control measure requirements
7. Create the initial hazard analysis
8. Update the requirements to include safety and reliability requirements

Thus the hazard analysis is an ongoing process that does not only include the design of a system but also its whole life cycle. Some parts of the analysis can only be done when a certain point of the design or development is reached, because preliminary information for these parts is not available before, see Figure 5.24).

Preliminary Hazard Analysis (PHA)

According to Leveson the goal of the PHA is to ensure that the conceptual design is safe, as flaws in the conceptual design that are discovered in later analysis stages can be impossible to fix [67] . Also it is often much easier and more efficient to ensure the safety of a system at the conceptual level. For example if one builds a car and then finds out that brakes are required, it will be very hard to integrate the brakes in the optimized car design and including brakes in the car concept and restart the development from that point will lead to a much more efficient and reliable solution.

Identification of Hazards

The assessment of the safety of a system begins with the identification of the hazards. This approach is supported by the extensions of the Hardware-Software-CoDesign formalisms developed in the project.
The analysis on the system level is hindered by missing long time studies on the behavior of airport personnel and aircraft crews to traffic signals the reliable indicate the status of a runway or a runway incursion alert. Based on the experience with ASDE-X and RWLY equipped airports the following assumptions are made.

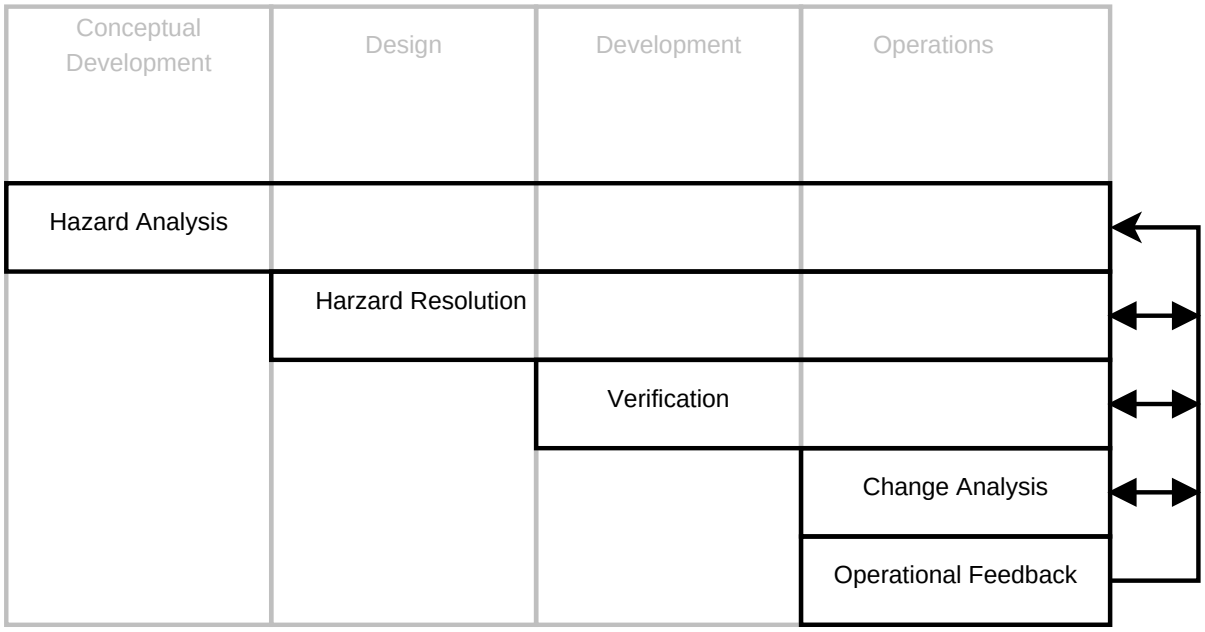


Figure 5.24: Stages of safety analysis mapped to a systems life cycle as in Leveson [67]

1. ATCO's rely on the alerting function of the XL-RIAS and react accordingly to a runway incursion alert.

2. Pilots also rely on the alerting function of the XL-RIAS and will react accordingly to the traffic signals.

Based on these assumptions the consequences of a failure of the XL-RIAS are determined. As with road traffic lights where a failure of a traffic light leads to a wrong decision and a possible accident of the traffic participants, a failure of the XL-RIAS is expected to result in a Pilot Deviation (PD), Operational Error (OE), or Vehicle/Pedestrian Deviation (VPD).
In the following the hazards of the XL-RIAS are classified into the classification scheme proposed by Douglass [27] which distinguishes between five fundamental classes. From these fundamental classes only the following three are relevant in the XL-RIAS context.

1. Release energy: The systems components are electrically connected with the airport infrastructure, therefore the system must not release harmful amounts of energy into the airport infrastructure. The system may be physically placed in an environment containing explosive liquids, such as aircraft fuel, in considerable amount. Therefore the system must be well insulated. The radar beam of the radar sensors must not interfere with other technology in the environment. Also exposed components like the radar may transmit energy absorbed from the environment like in case of a lightning strike.

2. Supplying misleading information to safety personnel or control systems: The XL-RIAS provides a alerting function, false alerts pose a significant tread to the safety of the airport.

3. Failure to alarm when hazardous conditions arises: If the system fails to alert in case of a runway incursion the risk of an accident is very high, especially once ATCOs, aircraft crews and ground personnel relies on the system.

In each of these categories the XL-RIAS poses multiple hazards to the environment [2]. These where identified and the following text provides a listing of the hazards in each category.

Hazardous release of energy: As a electrical system the XL-RIAS is most likely to release or transmit electrical energy that could damage the environment or cause secondary reactions that damage the environment.

- Release of hazardous amounts of electrical energy.
- Release of hazardous radar beam.
- Transmission of hazardous electrical energy.
- Transmission of hazardous physical energy.
- Not giving way if hit by something, like aircraft tires or a lawn mower.

Supplying misleading information to safety personnel or control systems: In case of a malfunction or tracking error the following failures of the system may follow:

- Failing to notify ATC / Aircraft crews in case of a system failure.
- False runway incursion alerts.
- False stop bar overrun warnings.
- Missing detection of vehicles in the area under surveillance.
- Inaccurate position information on vehicles in the area.
- Showing the wrong traffic signal.

Failure to alarm when hazardous conditions arises: To alarm in case of a hazardous situation is the primary purpose of the XL-RIAS. The inability of the system to alarm can be caused by failures of the systems components. If the system is not single-fault-tolerant every component failure will leave the system unable to raise an alarm in a hazardous situation.

- Failure to notify ATC in case of a runway incursion.
- Failure to notify ATC in case of a stop bar overrun.
- Failure to trigger the alerting traffic signals in case of a runway incursion.
- Failure to trigger the alerting traffic signals in case of a stop bar overrun.

[2]While the XL-RIAS is supposed to be a safety enhancing system, unreliability might result in a decrease in safety. Without the XL-RIAS the traffic participants an airport authority are responsible for the safety. Therefore it is to be shown that the XL-RIAS performs better.

Quantitative Hazard Assessment

Some of the identified hazards where countered by appropriate changes in the conceptual design, e.g.: The housing of the components had to comply to standards that have been proven to be appropriate for the environmental conditions encountered at airports. Other hazards are closely coupled to the reliability of the methods, algorithms and hardware used for the system. To assess these quantitatively the formalisms of the Hardware-Software-CoDesign where extended - details are discussed in Section 5.5.2 - to support the assessment of the reliability of the system design. This enables an approach different to Leveson hierarchical approach that requires dedicated System Hazard Analysis (SHA) ans Sub-System Hazard Analysis (SSHA). SHA and SSHA are accomplished in similar ways, but the goals are different, SSHA examines how individual component operation or failure affects the overall safety of the system, whereas SHA determines how normal and failure modes of the components operating together can affect system safety(see [67] pp. 153ff). In the unified Hardware-Software-CoDesign approach where system and subsystem hazard analysis is supported at the same stage of development, this distinction is not made. Another step recommended by Leveson is the Operating and Support Hazard Analysis (OSHA) that identifies risk reduction procedures during all phases of system use and maintenance. It especially examines hazards created by the Human Machine Interface (HMI). The OSHA has not been explicitly tackled during the project, but rather the standards based on the results of the OSHA of similar systems have been used to counter typical hazards.

Risk Assessment

The results of the quantitative hazard assessment were combined with the severity the outcome of a hazardous situation to assess the risks the XL-RIAS poses to the environment. The approach was to develop another extension of the Hardware-Software-CoDesign approach, based on an expert system that semi automatically analyzed the architecture graphs and the problem graphs to compute the effect of environmental influences on the system - e.g. a lawn mower overrunning a sensor - as well as intrinsic failure probabilities inherited from the nature of the components of the system - e.g. a ventilation failure leading to a emergency shut down of overheating components - and the resulting hazards - like over current and fire as a consequence of a lightning strike. While the basic algorithm was tested successfully on a limited set of events including fire, lightning strike, or physical demolition the database, that covers all reasonable incidents needed to compute a full analysis, could not be implemented during the project. Many risk are already minimized because the system needs to comply to various industrial standards that aim to minimize risks by enforcing the usage of interfaces, protocols and compliance to reliability and safety requirements.

5.5.2 Hardware-Software-CoDesign Extension

As a safety related system, safety and reliability played a major role during the design process. The Hardware-Software-CoDesign approach can be extended to support safety and reliability related engineering. The details of this approach are beyond the scope of this work, thus only a sketch of the approach is provided. The approach combines

elements of safety and reliability analysis into a integrated procedure and requires the following steps.

1. Extension of the formalism: The nodes in the architecture graph are extended to include numbers required for reliability and safety related calculations. These include variables like MTTF as well as maintenance intervals and repair durations. For nodes in the architecture graph all failure probabilities are marginalized into the following classes:

   The following set of states was chosen - based on the recommendations of IEC 61508 - to model the systems behavior.

   (a) Running no hw failure (RU) - In this state there a no problems, every part of the system is up and running and all safety critical functionality is available.

   (b) Running with undetected hardware failure (RUDF) - One or more components of the system have broken down but all safety critical functionality is available. The self diagnostic capabilities of the system have not notice the breakdown.

   (c) Running with detected hardware failure (RDF) - One or more components of the system have broken down but all safety critical functionality is available, but the self diagnostic functionality of the system noticed the breakdown and can initialized repair/maintenance actions.

   (d) Fail safe state (FS) - The system is broken down and the process that relies on the systems safety critical functionality is stopped.

   (e) Undetected failure (UF) - Due to hardware breakdowns unnoticed by the system the safety critical functionality is not fully available.

   (f) Detected failure (DF) - The systems safety critical functionality is not fully available but the system is aware of the problem. The system can initialized repair/maintenance actions and enter a fail save state if possible.

   Figure 1 shows the transition scheme for the availability states. The scheme is used during the generation of the markov models explained later in this section.

2. For a system problem graphs of safety related functions are generated. From the mathematical model the probability of a fault free execution of the problem graphs can be calculated using a Markov model generated from the problem graphs. This is the theoretic boundary for the reliability and safety of the system with perfect infallible hardware. The Markov models exploit the hierarchy of the architecture and problem graph by starting at the top level, constructing recursively Markov models for refined nodes. From the bottom of the recursion probabilities calculated from the model can be transferred to - and used by - the superior Markov models during the recursive ascend. Alternatively MTTF's for the refinements of architecture graph nodes can be estimated from the refined nodes components MTTF's assuming undetected failure, for the individual components and requiring all components to be running. This alternative approach always underestimates the reliability of the system because redundancy inherited from nodes in refined graphs is ignored.

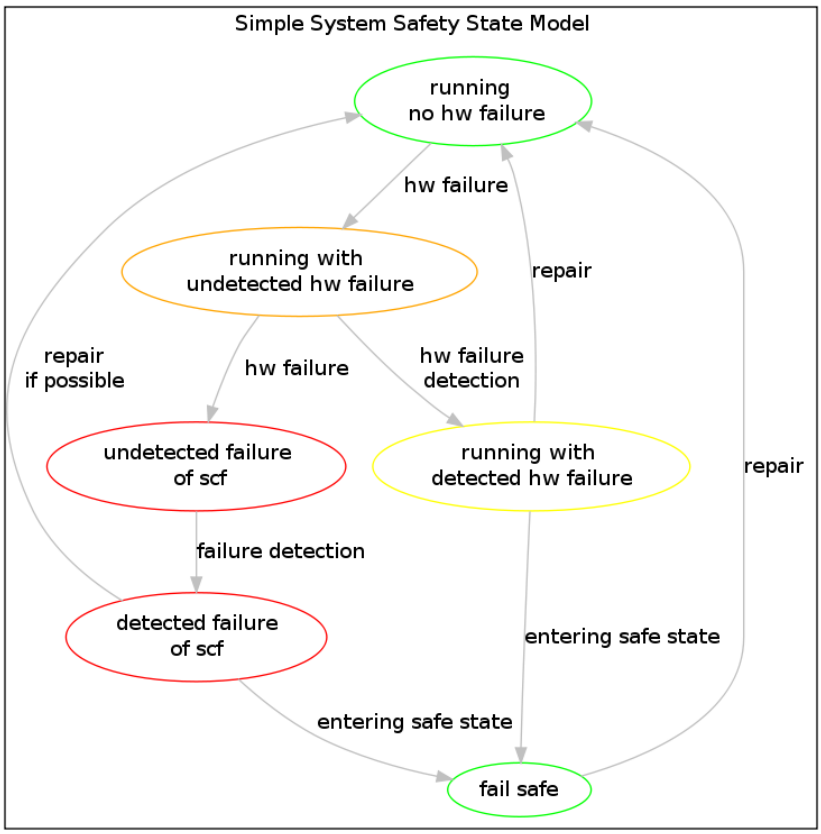


Figure 5.25: Transition scheme for architecture graph hardware availability states. These states are used to construct the Markov models for the Markov equilibrium analysis.

3. Based on the architecture graph nodes attached to each of the the problem graph nodes in the binding the chances for a hardware failure during the execution of the operations defined in the bound problem graph nodes are calculated.

4. A logic rule set performs a hazard analysis to estimate the probabilities of failure from risks beyond the immediate system boundaries. The probabilities of these failures can be integrated into the failure probabilities to build a compound failure model composed of system intrinsic and system extrinsic failure probabilities.

This approach allows the determination of the systems reliability and calculations of the probabilities for system failures. A subsequent hazard analysis allows the determination of the system's safety integrity level. In this scenario fail safe states play an important role. A fail safe state allows the system to recover from failure, or to be repaired while the operational processes that rely on the safety function of the system are on hold or the processes are moved to another part of the system. An example in the context of the XL-RIAS is that a failure of the sensor, that would compromise the systems safety function, would be signaled at the stop bar by turning on the stop bar and flashing the runway entrance light as an indication to vehicle crews as well as notifying ATC and GMC. The affected taxiway entrance would be closed and the traffic directed to other taxiway entrances. By deploying a set of safety related component monitoring functions

that put the system in a fail safe state, critical failures, such as losing the ability to detect runway incursions without notification, can be minimized. The final calculation of the system's state distribution over time and safety level is done by Markov model equilibrium state analysis, where the system is either up, down, or in fail safe mode. Figure 5.26 shows the system level availability analysis Markov model graph of a non redundant XL-RIAS design - see Section 5.6 for details - combined with the results of the equilibrium analysis. The model includes all states that have can be reached with a non negligible likelihood ($p > 10^{-}12$). Edges are color coded to highlight state transitions with significant likelihood ($p > 0.1$) and the size of Nodes codes the likelihood of the according state. The leftmost node marks the state where the system is up and running. Although the numbers are hardly readable at this scale the model graph shows that the system is most likely running ($p = 0.9999$) per hour. The remaining time ($p = 0.0001$) per hour the system is in on of the fail safe states and under repair. The blue edges from all fail safe state to the running state indicate that there is a significant likelihood for the system to be repaired.

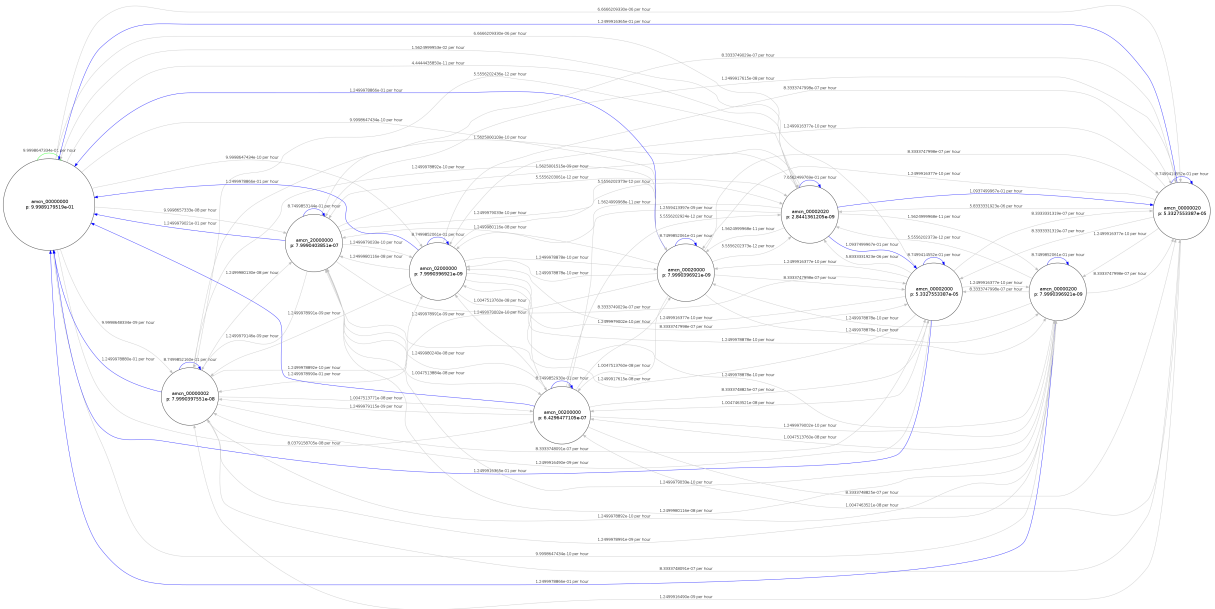


Figure 5.26: System level availability analysis Markov model graph of a non redundant XL-RIAS design.

5.5.3 SIL

To design a RIPAS that reaches at least the Safety Integrity Level (SIL) 1 was one of the research an project objectives. The estimates from the Hardware-Software-CoDesign extension developed during the project strongly indicate that the XL-RIAS - as defined in the following Section 5.6 - is satisfies the requirements of SIL 1 as a standalone design. The estimation is supported by the results from the alerting simulations study, see Section 5.4.

The analysis of the impact of the deployment of multiple XL-RIAS on a "dangerous" big airport like. e.g. Heathrow , that, in the opinion of many pilots, operates on an unaccept-

able safety level, using a Markov model of the airport landing and take-off operations - similar to the one introduced in Section 5.2.2 -found that it is possible to raise the airports safety to a commonly accepted level and reach SIL 1.

5.6 System Setup

Although the system is designed to locally track vehicles in a small areas of the airport surface, particularly hot spots, the system is design to incorporate arbitrary sensor measurements with ease and can be scaled for the reliable surveillance of ground traffic on a major airport, integrating sensor data from SMR, MLAT, ADS-B and other sensors into a single consistent view of the airports traffic situation. The following text's subject is the architecture of the surveillance of a single hot spot with proven of the shelf components, where the system is plugged into the existing airport infrastructure of a major airport. Figure 5.27 shows the supposed operational scenario of the system, where the system monitors the position of the aircraft and controls the local traffic signals according to the aircraft's movement and the clearances from air traffic control (ATC). The system level architecture of the XL-RIAS is shown in Figure 5.28. The nodes in the architecture graph are:

Figure 5.27: Lead on Lights at Dawn (Simulation)

1. Node Sensor:
   A functional architecture graph node. A sensor observing a hot spot area, generating sensor observations of objects in the area.

2. Node internal can bus:
   A communication architecture graph node. A CAN bus used for communication of the Local Intelligence Unit (LIU) with the sensors.

3. Node LIU:
   A functional architecture graph node. A Local Intelligence Unit (LIU). The local intelligence unit provides the data fusion functionality and has the responsibility to raise an alert in case of a runway incursion. The prototype LIU's architecture is similar to the one introduced in Chapter 3 and Chapter 4.

4. Node external can bus:
   A communication architecture graph node. A CAN bus used for communication of the LIU with the IO-Remote. The LIU receives information from ATC and feeds tracking data and alert signals to the AGLAS for distribution to ATC and traffic signals.

5. Node io.r.:
   A functional architecture graph node. An IO-Remote. The IO-Remote is a special interface to the AGLAS that is used to safe interface non AGLAS technology.

6. Node powerline:
   A communication architecture graph node. A power line that is used by the AGLAS to transmit internal information. The information is mainly used to monitor and control runway lighting and signals.

7. Node FRL:
   A functional architecture graph node. Fast Reaction Lights that form a traffic signal embedded in the runway/taxiway. The signal is set depending on the state of the system e.g. in case of an runway incursion alert the traffic signal might be flashing red.

8. Node AGLAS:
   A functional architecture graph node. An AGLAS with power line communication. The AGLAS is a system that allows the upgrade of airfield lighting to intelligent self monitoring LED technology by reusing existing infrastructure and serial power line circuits.

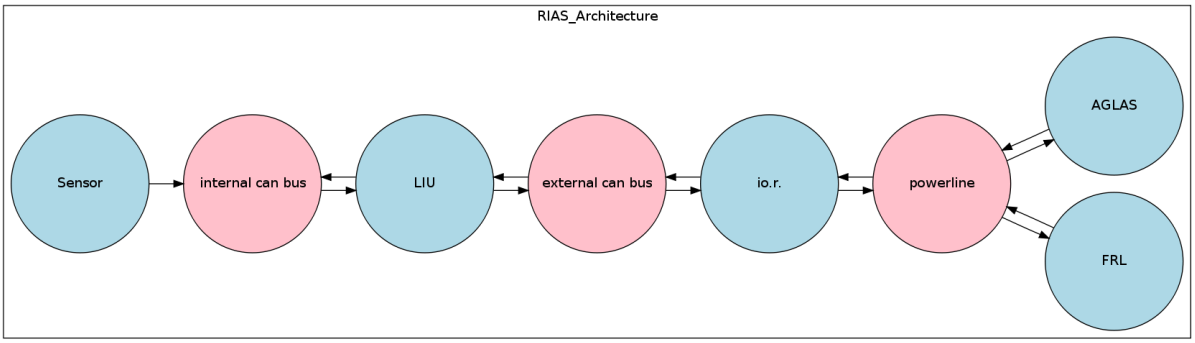


Figure 5.28: The architecture graph of the XL-RIAS on system level. For the explanation refer to the text.

5.6.1 Embedding into the Airport

The system may be embedded into the airport in many ways. It's primary purpose is to increase the runway incursion detection and alerting performance of existing solutions at hot spots. The performance of the systems already existing at the airport is boosted in such a way that they are able to handle a range of scenarios that cannot be handled

by this systems by now. But the system can also be deployed as a standalone solution that monitors the traffic on smaller airports or even untowered airports. The Hamburg Airport will serve as a demonstrator for this purpose. There are about 20 locations where the system would increase the safety of the airport. Each hot spot requires an unique sensor and signal configuration, so for the purpose of this explanation only 2 locations were chosen, see Figure 5.29. The locations were chosen because they are suited very well for the explanation due to the following reasons:

- Topographic Layout
- Visual Observability
- SMR Observation Quality

Topographic Layout: Hot spots are characterized by the local topography. A taxiway entrance is not automatically a hot spot. Additional topographic features make many taxiway entrances to hot spots.

Figure 5.29: Location A and Location B at the Hamburg Airport. The tower is app. 3km to the left.

- Location A: The location is a taxiway entry. The entry's topography forces aircraft that enter the runway, and wait at the hold line to face away from the runway threshold. Thus the crew can hardly see landing aircraft or might confuse aircraft taking off with landing aircraft, which is a cause for runway incursion incidents, see [90]. The hold line is very close to the runway shoulder, so there is almost no safety margin in case of a runway incursion and aircraft that are landing or taking off have difficulties to determine if the aircraft on the taxiway is behind the hold line or not.
- Location B: Here the taxiway entry is directly at the runway threshold, and the aircraft is facing the runway when positioned at the hold line. The disadvantageous topographic feature is the closeness of the hold line to the runway. As at Location A, here is almost no safety margin in case of a runway incursion.

Visual Observability: The visual observability of the location from the tower and starting or landing aircraft is also important for the risk in case of a runway incursion. If the visibility is not good, pilots have to rely on traffic information services and ATC for guidance. If these sources of information do not alert the parties in case of a runway

incursion, aircraft crews will lose valuable time for the execution of evasive maneuvers. In case of a false alert unnecessary evasive maneuvers pose a risk and disturb the airport operations.

Figure 5.30: Simulated visibility of Location A (left) and Location B (right) under adverse weather conditions from the view of an approaching aircraft. The aircraft at the taxiway is hardly visible.

- Location A: This location is too far away from the tower for accurate visual surveillance, especially under adverse environmental conditions. The location is also so far from the runway threshold that landing planes can't verify if the aircraft at the taxiway is waiting at the hold line or moving onto the runway until the landing aircraft has touched down. This makes the location extra risky since the landing aircraft is very restricted in its capability to perform evasive maneuvers and the aircraft on the taxiway is facing away from the landing aircraft and will most likely not notice the imminent danger. Therefore it will miss the opportunity for an evasive maneuver.
- Location B: The location is very far from the tower so a visual surveillance is not possible. The visibility from a landing aircraft is good, nevertheless the short distance between hold line and runway makes the distinction between an aircraft waiting at the hold line and one moving slowly onto the runway difficult.

SMR(i) Observation Quality: The quality of the sensor observations from the SMR are a very important factor as the SMR is, at many airports, the only global sensor capable of detecting non cooperative vehicles. The the big uncertainty of SMR measurements on the runway incursion detection poses a real problem because of the low update rate of app. 1Hz. The 1 second duration forces tracking algorithms to increase the uncertainty of the objects state between the measurements. This results in a suboptimal tracking performance that is not sufficient to reliable detect runway incursions.

- Location A: The location has a distance from the SMR that is frequently encountered at major airports and therefore is representative for many hot spots. The location is about 2.5 km from the main tower where the SMR(i) is positioned but has a clear line of sight. Due to the distance a visual verification of the position of aircraft positioned at the hold line of the taxiway entrance is not possible. The distance leads to a significant uncertainty of the sensor observations.

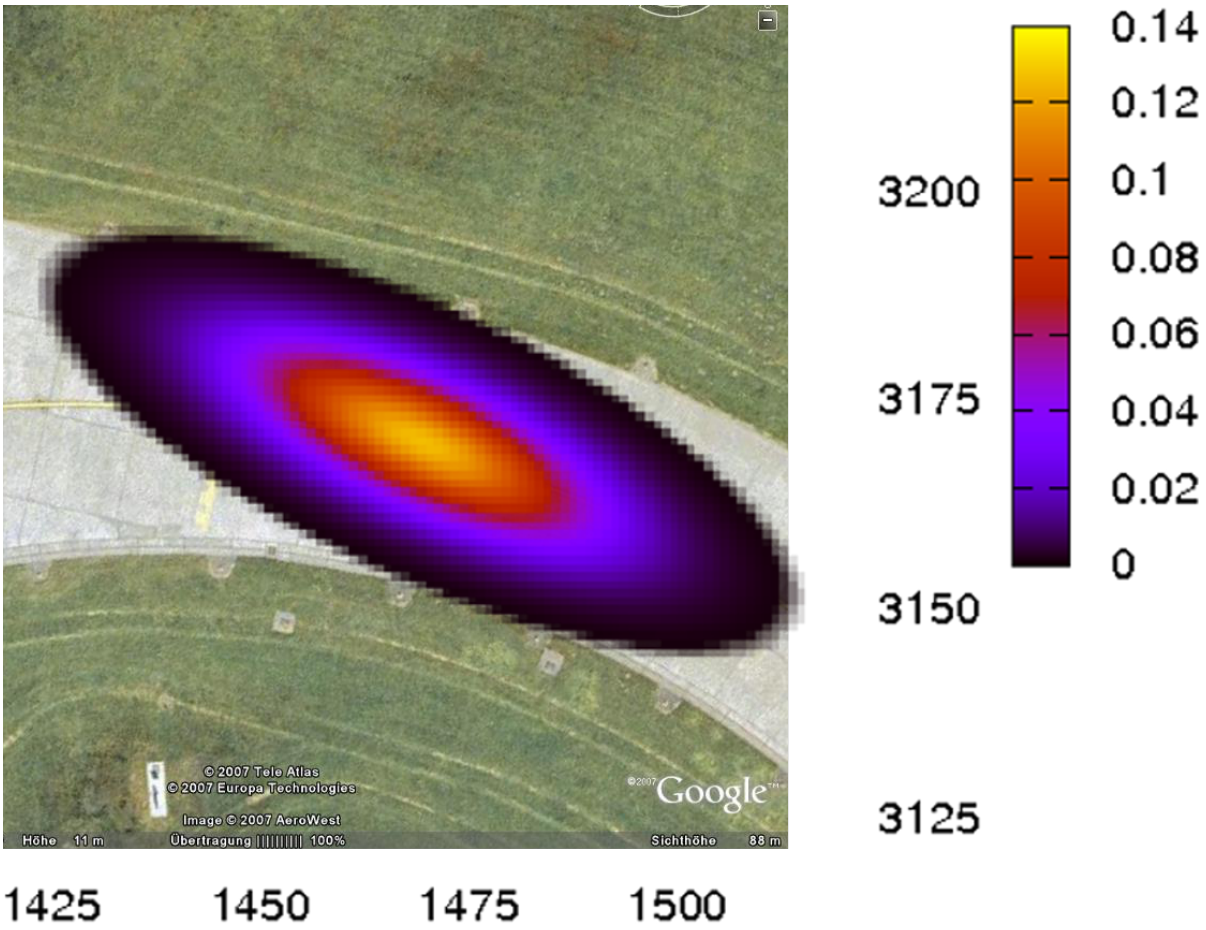


Figure 5.31: Overlay of the PDF of a SMR measurement at Location B. The numbers at the side denote the relative distance to the tower in meter. The color denotes the probability of a measurement at a certain area.

- Location B: The location is far away from the SMR which has a significant effect on the angular accuracy of the sensor observations. Figure 5.31 shows an overlay of the probability density function of a SMR observation at the taxiway entry. The overlay shows that it is not possible to accurately track a vehicle with the SMR at this location of the airport.

5.6.2 Sensors

The RIAS supports the inclusion of any sensor data that can be converted into the common representational format (CRF) within a sensible mathematical error range. These sensors include most currently available sensors commonly employed for airport surface monitoring. The position of the sensors in the architecture graph is shown in Figure 5.32.

Localized Sensors: As localized sensors universal medium range radars sensors from the automotive field have been chosen. Their characteristics according to the technical fact sheet of the producer have been validated by some examples and the characteristics have been used for the simulation. The sensors support a CAN bus and have been approved

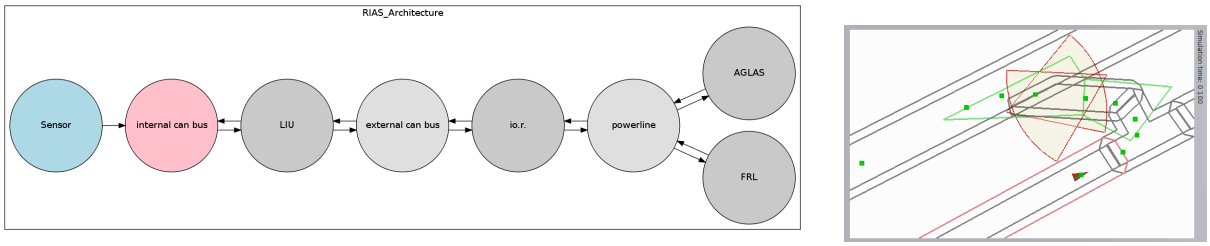


Figure 5.32: Left: Localization of the sensor in the architecture graph. Right: Setup of the sensors at Location A. The sensor beam area is marked by a reddish hue. The field of view of the third sensor is not shown.

for unrestricted usage by appropriate German governmental institutions. For the usage in the airport environment a few modifications are recommended, for example the inclusion of 3D acceleration sensors to detect collisions with aircraft or lawn mowers.

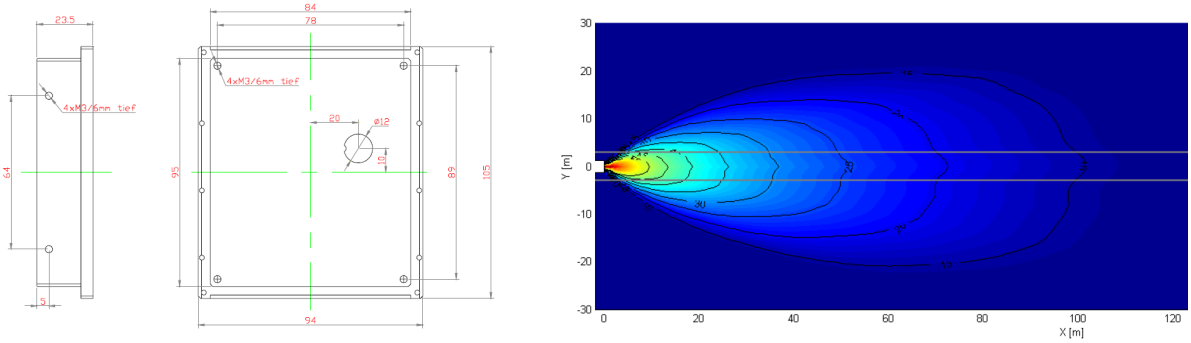


Figure 5.33: Radar sensor schematics: Left: The outdoor proven housing of the sensor is compact. Right: The field of view is sufficient for most surveillance applications.

Sensor Setup: The setup of the sensors depends on the topography of the location and the available infrastructure. In the optimal case multiple sensors cover the area in front of and behind the hold line. Competitive to the sensors used for tracking an additional sensor must be positioned directly at the hold line to ensure SIL 1+ requirements compliance. This additional sensor is positioned so that its broadest field of view axis is positioned vertically along the hold line. While this sensor has limited use for tracking it provides accuracy and reliability at the right place. Using an error correcting approach it is possible to detect unauthorized crossings of the hold line with very high reliabilities, the details will be explained in the following Section 5.6.5. The sensor setup is demonstrated at Location A. Figure 5.32 shows a schematic of the location including the sensors field of view, indicated by a reddish hue. The hold line is positioned at the intersection between the two green polygons. The runway is at the top of the image, the taxiway at the lower part. The taxiway entrance connects the runway and the taxiway.

5.6.3 Signals

To indicate alerts to pilots traffic signals made from groups of Fast Reaction Lights (FRLs) that are embedded in the runway and in the taxiway are used. These groups receive their control command via the AGLAS. The control commands are fed into the AGLAS by the LIU via the IO-Remote, see Figure 5.34 for the locations of the traffic signal control in the architecture graph. The setup of the signals is explained using both locations. Location B is used to explain the general setup, while Location A provides a vivid impression of the signal operation in case of a runway incursion.

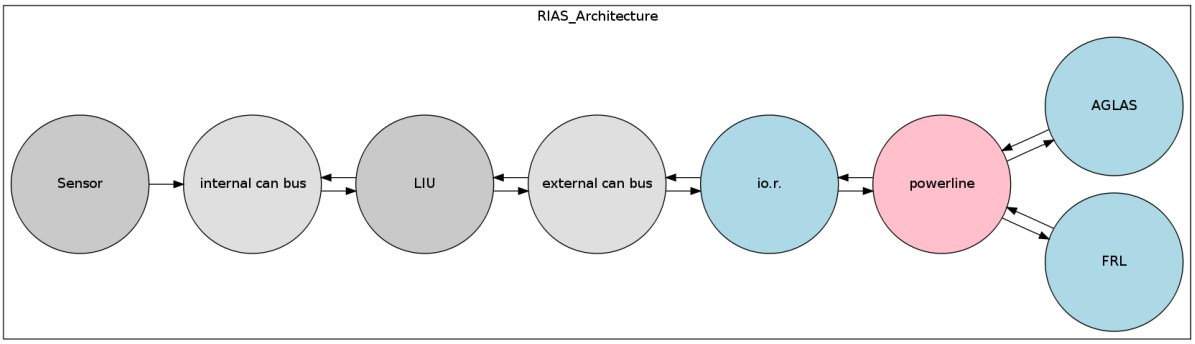


Figure 5.34: Components related to the control of the traffic signal in the architecture graph.

Embedding into the airport infrastructure: The traffic signals consists of groups of individual signal lights, as introduced above. The signals form logical groups that allow the usage of the same lights for different signals. Figure 5.35 shows the different signals at Location B.

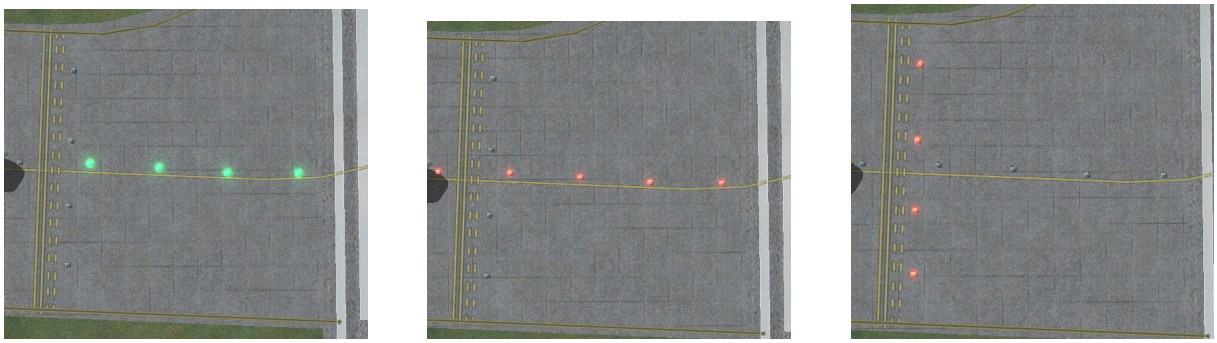


Figure 5.35: Setup of the signals in the Flight simulator environment. The setup consist of 9 intelligent multicolored LED Signal. The left picture shows the lead on signal that indicates that there is a clearance to enter the runway. The middle image shows the runway entrance lights that indicate that it is not safe to enter the runway, and clearance is canceled. The right picture shows the stop bar that indicates that there is no clearance given to enter the runway.

The first signal is the stop bar status, a hold line illuminated by red lights signals that the stop bar/ hold line is closed and their is no clearance given to proceed over the hold line

as in the right image on Figure 5.35. The lights are positioned at a distance of 3 m along the hold line, although there are indications that a distance on 1.5m might be better to prevent a blend with the runway lighting.
The second signal , the runway entrance light, is shown in the middle image in Figure 5.35. Red lights indicate that it is not safe to enter the runway because of a runway incursion. The lights run from the hold line along the taxiway center line to the runway center line.
The third signal is shown on the left image in Figure 5.35. The lead on lights indicate that an aircraft has explicit clearance and is requested to cross the hold line and proceed onto the runway. These lights also run from the hold line along the taxiway center line to the runway center line.

Operational Usage: Location A will serve to demonstrate the operational usage of the signals, the images in Figure 5.36 are taken from a demonstration video of a simulated runway incursion scenario. The first image shows the aircraft at the closed stop bar, indicated by the red signal lights. As the aircraft proceeds over the hold line a runway incursion alert is raised and the runway entrance lights are turned on. The signal is clearly visible to the crew of the aircraft and indicates that it is not safe to enter the runway and that a runway incursion just happened.

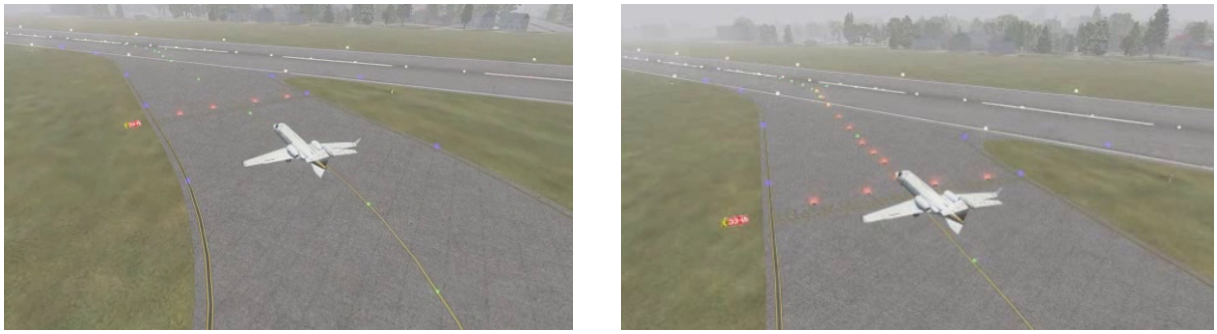


Figure 5.36: Operational embedding of the signals. Left: An illuminated hold line. Right: Red lights along the center line of the taxiway/runway indicate a runway incursion.

5.6.4 Data Fusion

The data fusion algorithms have been discussed in depth in the previous chapters. Here the realization of the architecture is discussed. Figure 5.38 shows schematic proposal of a possible setup of a single XL-RIAS. A key part for the embedding is the IO-Remote that is the physical and logical interface between the AGLAS system and the data fusion hardware and software or local intelligence unit (LIU). The IO-Remote makes it possible for arbitrary hardware to communicate with the AGLAS-Master via power line communication using a simple interface. During the course of the project the proposals for the integration of the LIU into the airport infrastructure has changed. In the beginning of the project the LIU was supposed to be placed in an underground service room near the hold line, later it was supposed that it would be best to integrate the LIU Hardware into the

IO-Remote. Through the integration into the IO-Remote the integration into the airport becomes much easier, as well as the maintenance.

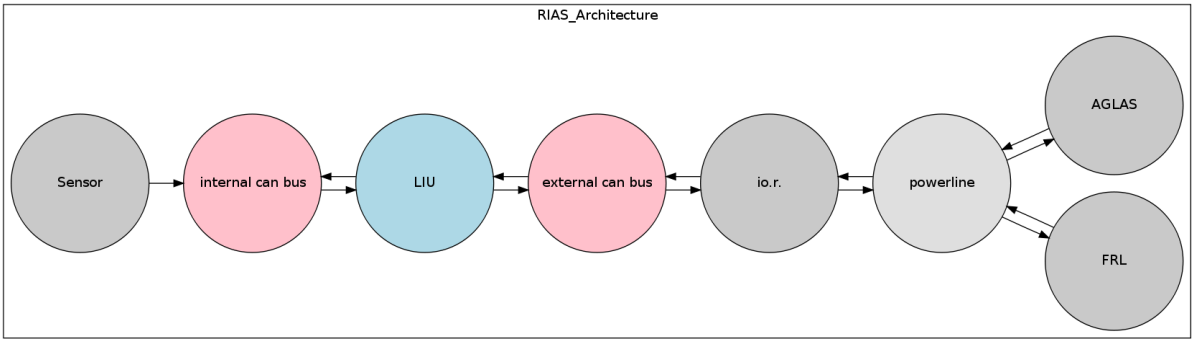


Figure 5.37: The components related to the data fusion process in the architecture graph at system level.

The reliability and safety analysis are also much easier as the analysis on the AGLAS system with respect to environmental and other external factors is already available and one can focus on the algorithms and mathematical methods. The integration into the IO-Remote has a direct influence on the actual hardware supposed for the realization of the design. Apart from the relative small volume available for the hardware, the standards of ADB apply some server constraints on the hardware. Some of the more important are:

- Environmental conditions: huge temperature fluctuations, lying into small pits together with current transformers and high voltage cabling, partially or complete immersed in dirty water containing a lot of chemicals
- Operating Temperature: - 40 to +75 °C
- Vibrations: 10g from 20 to 2000Hz
- MTBF > 100.000 hours
- Availability at least 99.99%

Although these constraints rule out most hardware the abstract architecture is not affected as much as one would expect. The proposed PC plus FPGA architecture does not create a too big heat dissipation, and the components are available for extreme environmental conditions. The availability of the non custom components of the IO-Remote require the usage of two or more IO-Remotes and failure resistant - with respect to the alerting context - signal control bus to ensure the safety integrity level of the system, but this does not affect the general architectural approach.

5.6.5 Additional Hold Line Surveillance

To further enhance the detection reliability of the system the third sensor is used to monitor the hold line. Because it's field of view is restricted to the area directly behind the hold line the sensor is immune to influences from outside this area. While this has

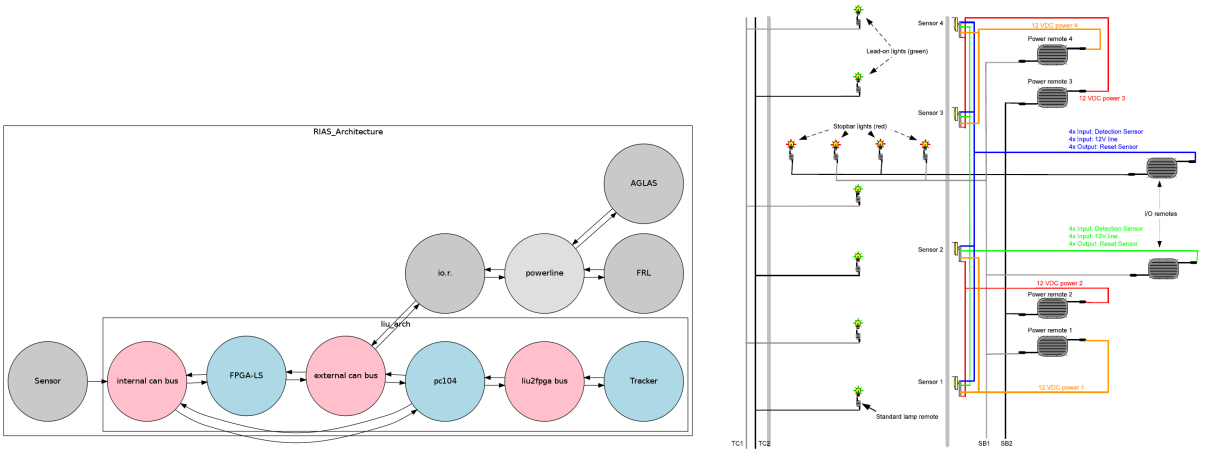


Figure 5.38: Left: Architecture graph with LIU refinement. The architecture has additional hardware for the independent line surveillance function performed by the additional sensor. Right: Integration of the data fusion as proposed by ADB during an intermediate stage of the RollMops-Project . The LIU is integrated into the IO-Remotes, four sensors are used for extra reliable hold line surveillance. The integration of the LIU into the IO-Remotes simplifies the integration into the airport infrastructure.

only a limited effect on the general tracking performance of the system it enables the independent detection of objects within the area directly behind the stop bar. This functionality is implemented in the FPGA-LS-Node in the refined architecture graph, see Figure 5.38
However a straightforward approach using a sliding window and the most likely cause for a series of measurements to determine if there is a vehicle or not is not going to work since the length of the sequence is bound by the maximum delay from begin of an event to system notification, which is supposed to be 100ms, and the sensor update rate is supposed to be 10ms-30ms which makes a maximum window size of ten measurements. The probability for a failure in determining if there is a vehicle in the sensor beam or not, using a window size from one up to 10 is 0.001 - 0.0001, which is not enough to build a SIL compliant system.
This limitation can be overcome by the use of an error correcting code, treating the measurements as a bit stream, and using a situation based model to define the words of the code.

Challenges

Because probabilistic methods are to some part model based the algorithms mentioned above rely on a sensor model and a state transition model. The sensor model is given by the specifications of the RIAS draft. The transition model is more complicated as it is given by the possible movement of a vehicle trough the sensor beam. It is of course possible to learn the transition model for each runway incursion area, but this would require a dedicated training setup or data of the airport traffic of the past.
The error correcting code also needs an alphabet and a framing procedure to work properly.

Proposal for a transition model: For the transition model one must deduce, by the size of the possible vehicles, their possible movements given the topography of the area and the characteristics of the vehicles and the sensor update rate, a transition model that is partly reasonable.
This transition model will be sufficient because the system do not need to detect what kind of vehicle is in the area, but just if there is a vehicle or a part of a vehicle in the area, and the model is sound for the first part of a vehicle as well.
The transition model assumes that a vehicle passing the sensor beam will cause at least n_v measurements and at most m_v measurements. That is because no vehicle is going to stay in the beam forever and every vehicle needs some minimal time to pass the sensor beam. The model further assumes that there is at least some space between two vehicles passing the sensor beam so there will be at least n_{nv} measurements where no vehicle is inside the sensor beam and at most m_{nv} since there is always a following vehicle.
The most interesting numbers are the measurements that are at least influenced, since they imply a transition probability of one to remain in the current state for the at least n_v or n_{nv} measurements, while assuming a $\frac{1}{3}$ transition probability to change the state for the measurements afterward until the next state change. This assumption is based on the fact that most of the time slow, big vehicles at a moderate traffic rate will influence the sensor, and not high speed traffic of small cars.

Using the transition model to build a code based model

The transition model also implies that a state change will cause a series of at least n_v and at least n_{nv} readings, and vice versa. This can be defined as a start and a stop sequence (message). Intermediate messages are any number of consecutive positive readings or consecutive negative readings. From this model a error correcting code can be derived.

Error correcting code approach The error correcting and error detecting properties of a code depend on the Hamming distance. The Hamming distance is the number of positions in which to codewords differ. To detect d errors a Hamming distance of $d + 1$ is needed. To correct d errors a Hamming distance of $2d + 1$ is needed. The Code has a number of words for vehicle and no vehicle of $m_v - \lfloor \frac{n_v}{2} \rfloor$ and $m_{mv} - \lfloor \frac{n_{nv}}{2} \rfloor$. The shortest words have $\lfloor \frac{n_v}{2} \rfloor$ and $\lfloor \frac{n_{nv}}{2} \rfloor$ digits. The system is not interested in the word itself but in the class of the word. The distance between the words of two classes is at least the length of the longer word since the word differ in every position. Assuming $n_v = n_{nv}$ and $m_n = m_{nv}$, there is always a word of the same length from the other class so the minimum distance between the two classes given any word is the length of the word.
The system can however not wait for the whole word. For this reason a message for a class consists of two words, the shortest word of the class followed by any other word of the class. Once the first word has been processed the system can assume that a vehicle has entered the sensor beam or has left the sensor beam.
The number of errors necessary to prevent the code from correcting the error is half the length of the word minus one.
The probability for such an event given the length of the codeword is

$$\sum_{n=0}^{\lceil \frac{length}{2}-1 \rceil} \binom{length}{n} 0.001^n \cdot 0.999^{length-n} \tag{5.1}$$

The minimum number of readings is assumed to be $n_v = 10 = n_{nv}$, so the length of the shortest words is 5 which is also the Hamming distance between the two first words of two classes. The probability that such a word is correctable is

$$\sum_{n=0}^{2} \binom{5}{n} 0.001^n \cdot 0.999^{5-n} = 0.99999999 \tag{5.2}$$

The probability that all start or stop messages in one hour are correctable depends on the maximum traffic rates which is assumed to be 500, 100 and 30 for example calculation. If all messages are correctable than no false detection will occur and no vehicle will be missed. These probabilities are:

- 0,99999002 for 500 vehicles per hour
- 0,999998 for 100 vehicles per hour
- 0,9999994 for 30 vehicles per hour

So the code as such is SIL 2 compliant for up to 30 vehicles per hour, and SIL 1 compliant for up to 500 vehicles per hour.
The challenge in this approach is the framing. The framing is the attempt to break up the stream of measurements into packets. In this case this can be done by start and stop flags signaling the beginning or end of a message. The state changes of the sensor beam can be used as start and stop flags, and can be detected using the Viterbi algorithm. An interesting property of the flags is that in this case the flags also contain information about the type of the word.
Given the conditional probabilities one can compute the likelihood for any sequence of length $\frac{n_{nv}+n_v}{2}$ to be a start or stop flag. The upper bound for the number of start and stop flags is given by the rate of traffic.
The probability to miss a flag when there is one is low, but not low enough. The hamming distance from a start to a stop flag and vice versa is in this case 10, the minimum distance to a intermediate sequences is 6 and the distance to continuous states is 5, so at least 2 errors are needed to miss a flag when there is one. The probability for less than 3 errors in a flag generating event is 0.99999045. That means that the system can satisfy SIL 1 requirements only if the traffic is no more than 40 vehicles per hour. The probability to detect a flag when there is no flag is somewhat higher. The chance for a sequence containing more than two errors has been estimated by simulation as about 0.3 per hour.The analysis of the long term equilibrium distribution resulted in a probability of 0.043 per hour for the appearance of a sequence containing more than two errors. The big difference results most likely from the fact that the simulation counts each appearance from 7 to 1 time, depending on the sequence of the errors. Assuming these probabilities are equally distributed to start and stop flags this leaves a probability of a detecting a false start flag of about 0.15-0.10 per hour.
To lower this probability one can choose a higher value for n_{nv} and n_v. The assumed sensor update rate sets the upper bound to be 20 because it implies a delay of 100ms

from the begin of the event to notification. The probability to miss a flag is now so low that the acceptable traffic rate is thousands of vehicles per hour. The hamming distance from flags to continuous states is now 10, so 5 or more errors in a window are necessary to cause a false flag detection. The probability for 5 or more errors in a sequence of 20 measurements is much lower than that for 2 or more errors in 10 measurements. The probability of detecting a false flag is then about 0.0001 per hour which is still to much. If one accepts only messages as flags that have a hamming distance of three to the flags. The hamming distance from such a sequence to a continuous state is now 7, and the resulting probability for a sequence containing 7 or more errors[3] is less than 0.000001. The probability to miss a flag is now still so low that the system can handle up to 2000 vehicles per hour while remaining SIL 1 compliant. Using this approach the system could satisfy SIL 1 requirements. In fact if one is only interested in the system state changes than just using a sliding window and treating consecutive flags as one, the flag detection can be implemented as a look up table , using the sequence in the sliding window as input.

5.7 Impact on the Runway Safety

The XL-RIAS is supposed to significantly enhance the runway safety of airports by means of runway incursion detection and alerting. The results from the performance simulation study show that the system can handle situations that are intractable by many currently deployed systems considered to be state of the art. The results of the safety and reliability analysis of the design strongly indicate that the system is compliant with SIL 1 requirements. Based on these results a mathematical estimation of the impact on the airport safety yielded results that indicate that the deployment of XL-RIAS at hot spots on airports significantly increases the safety of the airport. The system can reliable detect runway incursion and directly indicate the incident to the involved parties, thus reducing the probability of an accident. New unpublished results of a simulation study strongly indicate that the XL-RIAS can predict runway incursions even before they happen based on maneuver models appropriate for the movement of aircraft at runway entrances.
Besides the runway incursion detection and alerting capability the XL-RIAS has additional safety related benefits. Depending on the level of integration into the surveillance system of the airport the XL-RIAS can enhance the situation assessment of commonly deployed airport traffic surveillance system, by providing better estimates of the vehicle movements at hot spots and eliminate unreasonable movement estimations at hold lines where XL-RIAS are installed. The better assessment of the situation makes it easier for super ordinate instances like ATCOS or A-SMGCS to estimate the consequences that might arise from the traffic situation giving them more time to react in case of a runway incursion incident.

5.8 Prototype

A prototype has been realized for research and development. The formal architecture of the prototype is shown in Figure 5.39. The prototype features an additional camera for

[3] 7 errors or more are needed to transform a sequence of 20 consecutive positive or negative sensor readings to a sequence that is 3 errors away from a flag sequence.

visual inspection of sensor data. Camera images are recorded concurrently to the radar data. The video data allows the inspection of difficult cases during tracking. The CAN-Bus interface node and the Camera node can be moved to a separate PC in the local area network and transmit sensor data via Ethernet. Software trackers that substitute FPGAs to emulate the behavior when tracking multiple targets. There is also an interface to capture data from a flight simulator and trigger signals in the simulator. An additional GUI is available to interface the tracking system via network and visualize or control different aspects of the system. The prototype's design is more complex than the system design for a number of reasons which include but are not limited to:

- Greater Flexibility: The prototype XL-RIAS can be run with simulated or real input data. The simulated input enables experiments where ground truth is a necessity. Also comparative empirical studies that require exactly the same conditions are possible. Furthermore setups of multiple systems that are not possible because of limited resources and restricted access to airport facilities can be simulated. Because the FPGA tracker can replaced by a software simulating the FPGA's behavior, the effect of changes to the mathematical models and algorithms implemented in the FPGA tracker can be evaluated much faster than it would be possible with the FPGA only. The analysis of the dynamic development of the implemented particle filter is also better supported by the software simulation of the FPGA because the internal state is fully accessible.

- Desktop Development: The prototypes architecture allows to run the whole architecture on a single standard desktop PC including the simulated environment. Thus research and development is much faster and convenient than it would be possible if the prototype were an installed embedded system.

- Distributed Development: The prototypes software architecture allows the components to be distributed across a local area network. This way it is possible to position the sensors at a suitable location work in the office and keep the bulky AGLAS installation in a laboratory in the basement.

Software components

The formal architecture as in Figure 5.39 is distributed to a number of programs. The main software components are the LIU, the Fake FPGA, the LIU GUI, the RadarSensor-Interface, the Simulator and the FSX Interface.

LIU: The LIU program is a network of data fusion and interface nodes including the LIU-Software node, software trackers and interface nodes required to interface the simulated and real FPGA tracker. The programming language is C++.

Software FPGA Tracker: A small program that simulates the behavior of the real FPGA Tracker. Useful for testing, and debugging of the LIU and the FPGA algorithms.The programming language is C++.

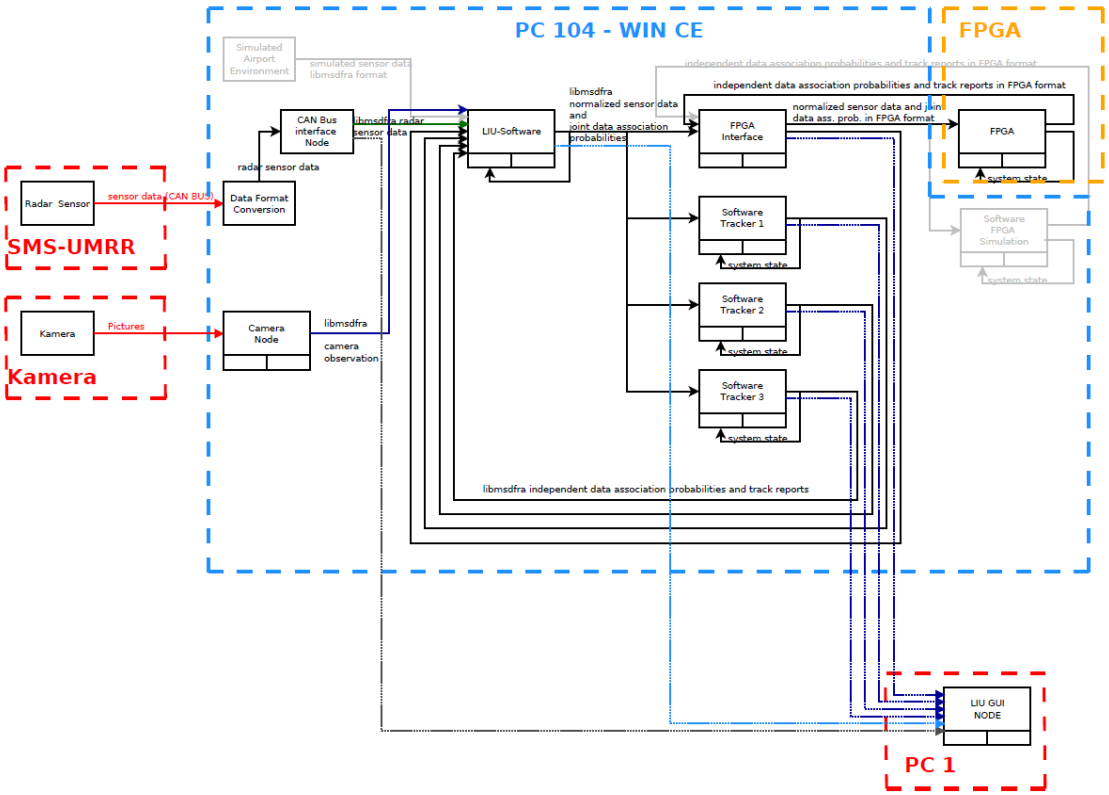


Figure 5.39: The formal architecture of the prototype, including a camera. All software parts can be run an industrial pc104 or a desktop PC. The gray parts are software substitutes for missing hardware and used for the tests of algorithms and interfaces.

LIU GUI: A tool for the visualization of the output of different data fusion nodes in the LIU program. The programming language is C++.

RadarSensorInterface: An interface to the radar sensor. Collects the observations that the sensor sends via CAN-Bus and provides a TCP-Server to distribute the collected data. The programming language is C#

Camera Interface: A camera is attached to the radar sensor to enable the visual inspection of the area under surveillance.

Simulator: Simulates sensor behaviors in a virtual environment. Microsoft Flight Simulator X (FSX) can be used as an input via the FSX Interface program. The simulated sensors output is accessible via TCP. The programming language is C++.

FSX Interface: An interface to the Microsoft Flight Simulator X, providing traffic data to the Simulator program. The traffic is detected by simulated sensors in the simulator. The programming language is C#. An interface were information can be feed back to the

simulator to control simulated traffic lights was developed but has not been integrated in the prototype by now.

5.8.1 FPGA Tracker Prototype

The FPGA prototype implementation of the particle filter, as discussed in detail in Chapter 4, has been done on an "Altera Cyclone III Nios II Embedded Evaluation Kit" with a 50MHz Clock. For peripheral interfacing the NIOS II soft core CPU with an Ethernet interface has been used, consuming about 60% of the 25.000 LEs on the FPGA. The NIOS II connected the prototype via Ethernet to the superior software. The actual particle filter logic (PFL) reads has been implemented using the remaining 40% of the available LEs. The PFL reads and writes the data to the NIOS II via custom bus system. The NIOS II communicates with the rest of the tracking system via Ethernet.

5.9 Summary

In this chapter it is shown how the embedded architecture introduced in the preceding chapters can be used to design a system that is applicable to a significant real world problem, the prevention of runway incursions. The systems performance in the context of the airport environment in front of the background of the detection and prevention of runway incursion has been examined in detail and compared to existing solutions. The impact on the airport safety has been evaluated using an extension of Hardware-Software-CoDesign formalisms and mathematical analysis of the airports landing and take-off operations with respect to the detection and prevention of runway incursions. The feasibility of the realization of the design has been demonstrated and a prototype has been realized.

Chapter 6

Conclusion

Contents

6.1 Summary

In this work an architecture for embedded multi sensor data fusion is proposed that allows the application of tracking algorithms in scenarios that do not allow for software solutions, because of constraints that apply from the embedding context. Especially tracking applications in the areas of airport ground traffic surveillance, autonomous driving, driver assistance technology, small autonomous robots, and indoor pedestrian tracking require an efficient reliable hardware tracking solution. In front of this background research goals were defined with respect to the state of research in the affected fields, and a good deal of them has been reached.
The general objective of the integration of the support of domain specific concepts into the hardware has been partially reached, the tracking hardware developed in this thesis supports the usage of domain specific state transition probability density functions as well as domain specific sensor characteristics. These can be transmitted to the hardware in addition to sensor data to enhance the performance in terms of accuracy of the results computed by the hardware. The result has been reached by deriving suitable general applicable mathematical models that allow the usage of domain specific concepts such

as the movement of a vehicle through the parametrization of the mathematical models. Domain specific concepts are further supported through the modular formal architecture and its VHDL implementation, which allow to change parts of the computation without the necessity to change the whole formal model or VHDL implementation. The objectives related to the more specific research topics embedded tracking, RoboCup, and airport traffic surveillance have been reached almost completely. The following text provides a concise listing of the status of the objectives.

6.1.1 Embedded Tracking

Embedded Tracking - Objective 1:

The objective has been reached. A hardware architecture for a particle filter based tracking system was developed. A working prototype has been realized as a proof of concept. However the formal framework for the semiautomatic parametrization of the hardware for common application scenarios could not be finished in time.

Embedded Tracking - Objective 2:

A distributed hardware software architecture for tracking of multiple objects has been developed. The architecture supports the embedding of the particle filter based tracking system into superordinate systems from the robotic, automotive, and airport domain. The system is made up of a template based C++ library.

Embedded Tracking - Objective 3:

As part of the distributed hardware software architecture a formal interface for easy usage of the hardware has been developed. The formal interface allows the convenient integration of the hardware in superordinate hardware and software systems. The provision of a Ethernet interface by the hardware and the formal communication interface allow the seamless integration into a wide range of systems using common programming languages like Java, Python, C++, or LISP.

6.1.2 Airport Ground Traffic Surveillance

Airport - Objective 1:

It has been shown that it is feasible to build a runway incursion alerting system based on localize sensors and localized signals. Experience from ASDE-X and RWLS systems installed by the FAA on major airports in the USA strongly support the conceptual design of the system. It has been shown that runway incursion alerting systems based local surveillance can handle a wide range of runway incursion scenarios but need additional data sources to incorporate information about landing aircraft to provide full cover of reasonable runway incursion scenarios. It has been shown that it is possible to push the functional safety level of airport taxi, landing, and take off operations with respect to runway incursions to a safety integrity level (SIL) of at least SIL 1.

Airport - Objective 2:

Show the superior performance of a runway incursion system based on localized surveillance. A comparative simulation study based on the performance of sensors from the automotive field and off the shelf components showed that it is possible to significantly boost the performance of systems that strongly depend on global sensors by the integration of local sensors. The significance of the performance boost is shown by the range of new runway incursion scenarios that can be handled. The performance has been demonstrated by applying the system to real runway incursion incidents that happened in the past.

Airport - Objective 3:

The superior performance encouraged the development of a design for a local runway incursion alerting system based on local sensors and off the shelf components. The design combines a tracking and a line surveillance approach to achieve optimal reliability and maximum safety through a synergic effect by the fusion of the approaches on the level of object assessment.

Airport - Objective 4:

A prototype of the system design using the particle filter based tracking system has been almost completely developed for the airport environment. At the time of writing the prototype is still missing a component for the interfacing of the signals to be provided by the industrial project partners. The prototypes hardware architecture is - as the focus is on research and evaluation - slightly different from the system design, which is targeted on the airport environment.

6.1.3 RoboCup

The objective to integrate the particle filter based tracking system into the ball tracking of the RoboCup Software and demonstrate the performance gain could not be reached due to missing time. Preparations for the integration and the simulator test are almost done, but the test hardware in the loop test framework is not finished and it would require some additional months to execute all experiments and evaluate the experimental results. To finish this part of the work it has been moved to future work.

6.2 Contribution to the State of Research

After the end of the non-disclosure agreements with the industrial partners, contributions to the state of art during the course of the research work were made in the form of conference papers and an article [99, 98, 97]. These works partially publish research results obtained during the course of the research. Further contributions are made by results published in this thesis. Parts of the previously published results are included in this work for the sake of a complete thesis.
In the editorial of the march 2012 issue of the ICGA journal - titled "The Communication of Ideas" - Jaap van den Herik stated that "Progress in science is based on the development

of ideas" and stresses the plain communication of ideas. A PhD-thesis is usually not the plain communication of an idea - opposed to an article or a conference paper - but rather the explicit examination of an idea, often more than just one idea, which does not only explain the idea but also demonstrates their value. This thesis provides the necessary framework to show the value of various ideas related to the research objectives.
In this subsection I will explain the new ideas contributed to the pool of proven useful ideas and what of the already existing ideas I could extend to apply them to areas where they have not been applied before. In addition to the results already published [97, 98, 99] this thesis makes contributions to the state of research with respect to the theory and praxis of embedded tracking system architecture , safety oriented Hardware-Software-Co-Design, and modeling reliability and safety of runway incursion systems. The following list of the contributions provides an overview on the ideas and insights that contribute to the state of research - beginning with those already published.

1. Effective zero weight treatment for hardware particle filters, already published in the proceedings of the Mathmod 2012 [98].

2. It has been shown that general centralized traffic surveillance solutions do not provide the necessary accuracy and reliability to implement safety measures against a particular dangerous kind of runway incursions. This result has been published partially in an article [99].

3. It has been shown that it is possible to increase runway safety, through the use of local sensors and signals using embedded data fusion algorithm, already published in proceedings of the IEEE-MFI 2012 [97].

4. A new extension of hardware-software-co-design formalisms to support semiautomatic safety and reliability analysis.

5. It has been shown that it is reasonable and possible to significantly increase the performance of small autonomous robots using embedded tracking algorithms with respect to power consumption and heat dissipation.

6.3 Directions for Future Work

The research done has answered a few questions but ,as usual, raised more. The decision which of these should be in the focus of the future research did prove as challenging as the central research of the thesis. As for the thesis, not only scientific topics but also engineering topics are relevant for future work. Many of the scientific ideas formulated in this work are off limited use without the embedding in an engineering context, this is particularly true for the mathematical models, formal architecture, and algorithms that provide a new scientific solution that allow a better solution of an engineering problem.

6.3.1 Embedded Tracking Systems

The architecture for the embedded tracking system and its implementation developed in this work proof the concepts, ideas, and the feasibility of a real world realization.

On the engineering side what remains open is A) to bundle all the insights gained and the experiments made into a formal and software framework that allows the convenient, semiautomatic integration of the hardware into application scenarios, B) to optimize some aspects of the hardware to increase the performance of the design, and C) round up the implementation with more formal models to choose from.
On the scientific side there are numerous challenges regarding the better support of domain specific concepts and background knowledge in an architecture based on concurrent computation. The further integration of these enhances the performance of the tracking system in scientific terms, such as the error or probability of convergence.

6.3.2 Automotive

Currently the automotive field is about to face of a new generation of cars. The development of self driving cars have made enormous progress during the last years. Google's autonomous cars have so far traveled over 300.000km under very challenging conditions without an accident. In 2013 autonomous cars are allowed in 3 states of the USA for driver license tests. Less ambitious but more common are driver assistance systems that aim to prevent accidents or at least minimize the harm from accidents by providing enhanced situational awareness and even actively taking over the control of the vehicle in case of a missing reaction from the driver. Currently deployed hardware and software does not support the integration of multiple sensor data across cars to build a more complete and accurate model of the surrounding of a group of cars. The developed hardware and software supports this kind of fusion that allows a better estimation of traffic situations. Also commonly used alpha-beta filters and Kalman-filters can hardly predict the outcome of a multiple vehicle interaction for more than a few seconds. The particle filters intrinsic ability to model arbitrary probability distributions allow the prediction of the probability distribution over the possible courses of a multi-vehicle interaction. With this information a better and coordinated planning of car actions will become possible that is based on a multitude of possible futures. However these solutions need to be realized as hardware in order to be optimally suited for the automotive field. What does work for an autonomous test car does not work for every day usage. This applies mainly to functional safety and reliability, where software solutions do usually not meet the demands, and the computational complexity of the data fusion algorithms does not allow a physical form as compact as desired, unless done in hardware.

6.3.3 Robotics

Future work in the robotics domain is closely coupled to the future work regarding embedded tracking systems. Without any further progress in the research on the embedded tracking system the work will focus on the task to integrate a prototype into a robot from the RoboCup humanoid soccer league for tracking teammates, opponents, the ball, and possible obstacles. The performance of the robot with the tracking hardware would than be compared to that of a robot without tracking hardware. A scientific challenge is to extend the concept to track individual primitive features from monocular video streams and derive the semantics of the tracked primitives using Bayesian inference. A similar approach based on machine learning [108] and Kalman filters[89]to establish semantics

has been published lately.

6.3.4 Airport

During the works required to gather the necessary background knowledge for the development of a runway incursion system, see airport traffic surveillance objectives, it became clear that the currently deployed systems do not provide full cover of the most dangerous runway incursion scenarios for various reasons. The nature of commercial airport operations demand solutions that satisfy the needs of multiple industrial standards while being affordable. Complex system architectures for local surveillance, that do not seamlessly fit into existing infrastructure, are therefore often not even considered for deployment. The complexity of the system architectures is mostly due to the computational demands of sophisticated data fusion algorithms, particularly those needed for tracking and image processing. Therefore the integration of the developed system design into a commercial hardware, such as the IO-Remote by ADB, or sensors would help to provide local runway incursion technology that satisfies the needs of airport stakeholders and provides the necessary safety demanded by governmental institutions like the NTSB. Adding better support for domain specific concepts, like map based movement stochastics, to the embedded particle filter will enhance the tracking accuracy of the hardware.

Appendix A

Standard Key Technologies

For almost every part of the RIPAS in Section 5.2, implementations are available. In RIPAS systems, different components for the same task are often used concurrently to achieve a better performance of the overall system.

A.1 Traffic Information Services

Traffic Information Services (TIS) provide local traffic information for the participating users. The reliability of the technology depends on the topographical circumstances and the setup of the antennas at the airport as well as on the aircraft systems. During runway incursion prevention system tests in 2000, it was found that the failures of the system to alert the pilots in time were caused by unreliability of the Automatic Dependency Surveillance-Broadcast (ADS-B) and Traffic Information Service Broadcast (TIS-B) traffic data [9, 119, 10, 59]. The update rate of the traffic information also depends on the topographical circumstances and the antenna setup, and has been found to drop down to 0.1 Hz in field testing [57].

TIS-B Traffic Information Service Broadcast (STIS-B) is a service that transmits airport surface traffic data. It can be viewed as the ground to air communication of the Adaptive Dependency Surveillance-Broadcast. This service could suffer from inconsistent timing of the traffic data, as found at Dallas/Fort Worth International Airport (DFW); also, such systems could provide unfiltered data from all of the sensors that must be filtered by data fusion algorithms to avoid false alarms, as encountered at DFW [59].

ADS-B Automatic Dependency Surveillance-Broadcast (ADS-B) is a surveillance technology for tracking aircraft as part of the Next Generation Air Transportation System (NextGen) [82] The United States will require the majority of aircraft that operate within its airspace to be equipped with some form of ADS-B by January 1, 2020 [82]. ADS-B messages usually include position, altitude, speed, and heading. The usual update rate is approximately 2 Hz [57]. For General Aviation (GA), the message accuracy is often not as good as that of modern airliners because it is not smoothed by inertial sensor data.

A.2 Traffic Surveillance Sensors

Airport ground traffic movements can be assessed in various ways, surveillance sensors monitor the position and movement of vehicles through different means. One can generally distinguish between sensors that are part of the vehicles and sensors that are part of the airport infrastructure. There are sensors in between these two polar points that require an active part on both the vehicle and the airport infrastructure.

Vehicle Sensors

Vehicle sensors are sensors that are part of the vehicle. Most aircraft today have a broad range of sensors, which can be roughly classified into GPS, inertial sensors, and extrinsic sensors.

1. GPS: The Global Positioning System (GPS) is a satellite-based positioning system with an accuracy that is often better than 10 m. The accuracy can be enhanced by the use of a Satellite Based Augmentation System (SBAS), to approximately 1 m. DGPS uses ground-based reference stations to estimate a correction factor and can achieve an accuracy of 0.3 m- 2 m. The heading of a vehicle can be determined with an accuracy of 0.01 degrees to 0.1 degrees.

2. Inertial Sensor: Accelerometers, gyroscopes and odometers can be used to estimate the movement of vehicles with high update rates and to interpolate and smooth the position estimation given by the GPS. The resulting accuracy is much higher than that of the GPS-only based position estimation, and a reliable prediction of the movement of the vehicle is possible.

3. Extrinsic Sensors: This class of sensors measures some properties of the environment of a vehicle. Because sensors that assist driving are already successfully in use in the automotive field, the same class of sensors can be used to enhance the situational awareness of pilots and vehicle operators. These sensors can detect obstacles such as pedestrians or vehicles at a reasonable range to avoid collisions during taxiway operations.

Airport Sensors

Airport sensors are part of the airport infrastructure. One of the most important properties required by the sensors is that they work under challenging environmental conditions, for example, when the visibility of pilots and Air Traffic Control (ATC) is limited. However, even sensors that are suited only for clear weather could prove to be useful because field studies found that the visual 'out of the window' detection of runway incursions often fails or lags behind automated alerts even during the best weather conditions [104].

1. Surface Movement Radar (SMR): A global and usually non-cooperative sensor that detects vehicles over the whole airport area. Its update rate is usually approximately 1 Hz. Its capabilities for locating smaller vehicles and pedestrians are limited. Even high performance systems such as Airport Surface Detection Equipment Model 3(ASDE-3) radar and Airport Surface Detection Equipment - Model X (ASDE-X) radar suffer from Multi path reflections that can lead to false target reports [57].

2. Low Cost Surface Movement Radar (LCSMR): These mill metric radars are available with resolutions of 0.25 m and with ranges of 800 m and 360 degree coverage. Update rates are approximately 1 Hz. By deploying these radars at strategic places in the airport, full coverage can be achieved. The systems are usually small and can be installed near the ground (2 m height) [1].

3. Universal Medium Range Radar (UMMR): This limited range radar has high spatial and time resolution. It is often used in automotive applications. Its resolution allows the reliable detection of targets as small as pedestrians.

4. Multilateration Mode S Radar (MLAT): Used as SSR, this type of radar requires the target to wear an active transponder. The position of the target is calculated from the signal run times to different receiver stations. The signal can also be used to establish a data link to transmit information such as speed. The update rate is usually 1 Hz, but special localized systems can reach up to 6 Hz through the use of active transmitters. In practice, transponders are often not used by flight crew and ground traffic [62]. Some MLAT systems (installed prior to 2005) may receive but not distinguish between ADS-B and TIS-B messages, resulting in a malfunction of the tracking algorithms [110].

5. Microphone Arrays: Experimental systems that identify and locate aircraft by their sound. The localization quality has been found to be quite high in experimental field studies. Sound signatures also allow the resolution of the aircraft type.

6. Thermal Cameras: Thermal cameras are used to detect planes by their thermal signature. Field studies found these cameras to be useful for the surveillance of larger aircraft.

7. Cameras: Used for the surveillance of hot spots, localized high resolution high speed cameras can be used to identify and locate aircraft accurately but suffer from disadvantageous environmental conditions. Some systems use a supplementary thermal camera to address some of these problems.

8. Microwave Barrier: Used for the surveillance of stop bars, this type of sensor has been found to be quite reliable and even allows for the detection of vehicle types, according to a field study [20]. Microwave sensors have also been used in a EUROCONTROL study [86].

9. Inductive Loops: Used for the surveillance of stop bars, inductive loops have been found to be unreliable by some authors [75] and reliable by others [28].

10. Magnetic Field Sensors: These sensors are experimental sensors that use the deformation of the earth's magnetic field to locate and identify planes [23]

11. Laser scanner: Laser scanners provide detailed information but are quite expensive and suffer in performance under challenging environmental conditions; thus, they are not the first choice for runway incursion detection. However, recent advances in the development of all-weather infrastructure and automotive laser scanner technology could change this limitation.

A shortcoming of many sensors is that the uncertainty with respect to the accuracy, probability of detection, and false alarm probability is that it changes with the location of the vehicle, or even with the constellation of the vehicles at the airport. A good example is the MLAT performance, that is very well documented in the reports [77, 78], and gives a vivid impression of the qualitative differences between various locations. Figure A.1 shows plots of the uncertainty of MLAT measurement. Most data fusion systems do not integrate the local uncertainty, which leads to a false estimation of the accuracy of the tracking and suboptimal tracking performance.

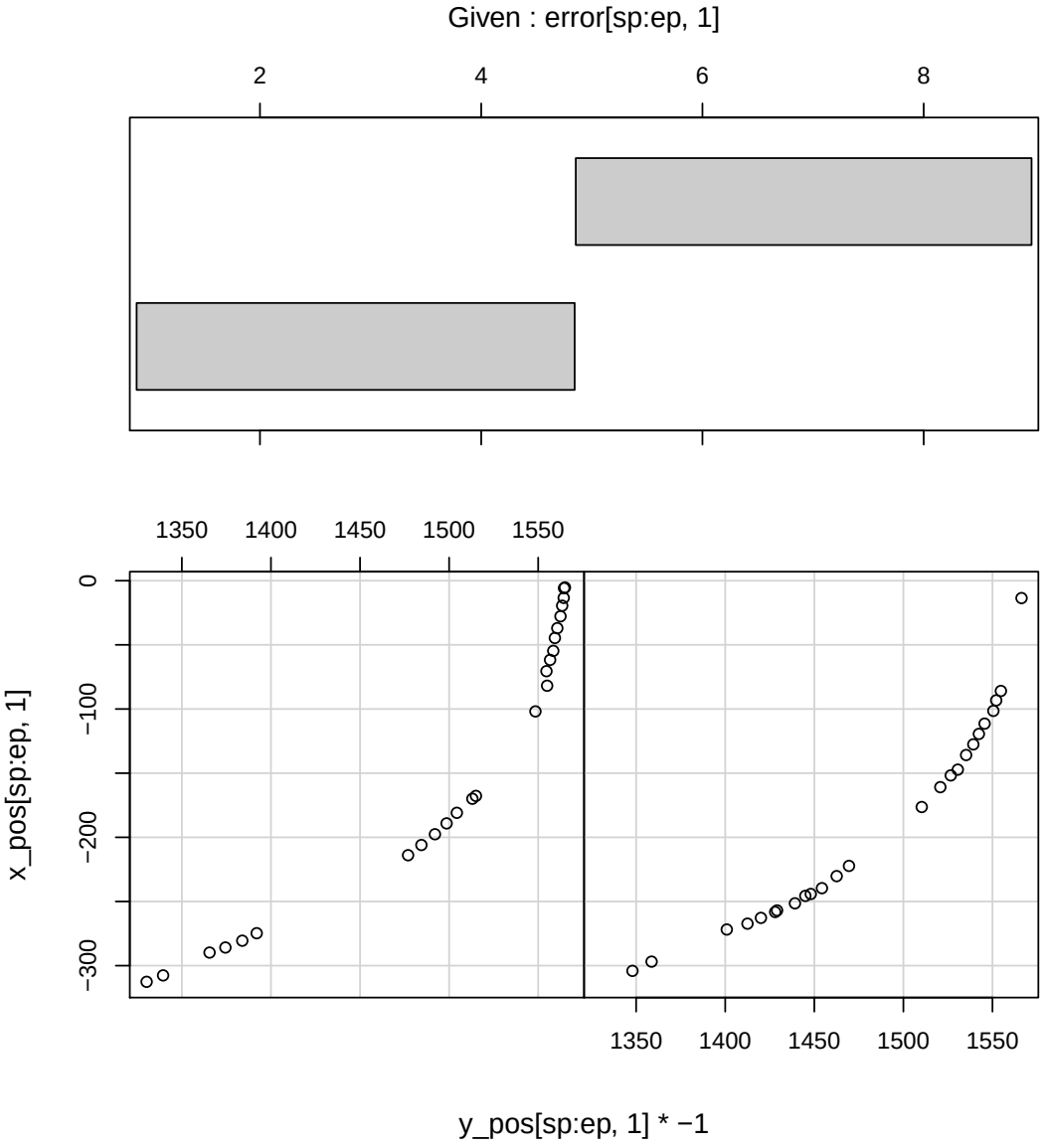


Figure A.1: Local uncertainty values of MLAT measurements. The plots show that, despite the good overall performance of the MLAT, there are systematic errors that appear to be directly related to the location of the detected vehicle on the runway. The data was taken near an taxiway runway intersection.

A.3 Human Machine Interfaces

Pilots and ATCOs have different needs for Human Machine Interfaces (HMI's). While Pilots mainly depend on the output of the HMI, ATCOs also need the Human Machine Interface (HMI) as an input device.

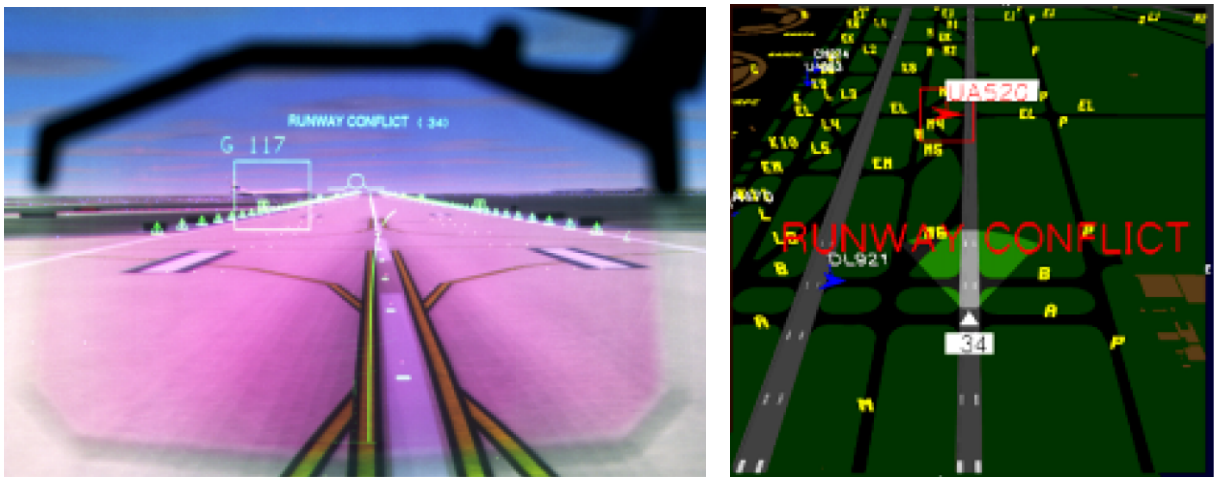


Figure A.2: HMI Visual Alerts. Left: Visual alert in a Head-Up Display (HUD). Right: Visual alert in an Electronic Moving Map (EMM). Because the pilot's attention may not be focused on the EMM, audible alerts have been found to decrease the reaction time. Images are reprinted from NASA report [13].

Flight Crew HMI

HMIs for pilots must be integrated into the cockpit of aircraft. In the opinion of the pilots, runway safety HMIs are valuable but increase the complexity of the cockpit. Therefore, most pilots advised to integrate the HMI into multi-function displays (MFD), to reduce the complexity of the cockpit and to allow larger displays [109, 54].

- EMM: Electronic Moving Map: The electronic moving map is a Map that shows the airport and the traffic around the plane. This map can be set to different modes, displaying the map from different viewpoints. Routing information can be integrated to show the route on the display. The plane is also shown. Depending on the operational state of the aircraft, pilots favor different views on the maps; for taxiing, top down views and slightly perspective views are favored, while for landing operations, a perspective view from the aircraft to the runway is favored.
- IMM: Integrated Moving Map: Roughly the EUROCONTROL version of the Federal Aviation Administration (FAA)/ National Aeronautics and Space Administration (NASA) EMM[109].
- Audible Alert: Because the attention of both the pilot and co-pilot could be focused on things other than the display's runway incursion warnings and alerts, audible signals have been found to be of great value in various studies [119, 109].

- HUD: Heads-up displays project valuable information into the field of vision of pilots. Directions of converging traffic can be shown, and vehicles can be marked, to draw the attention of the pilot to dangerous situations. Additional information on aircraft and vehicles, such as speed, can also be shown in the HUD. This capability could provide vital information because it has been found in runway incursion incident investigations that pilots of landing aircraft experience slow moving interfering traffic as being non-moving [86].

- RTS: Radio telephony can be used by airport-based runway prevention systems, to transmit warnings and to provide alerts that are audible to pilots. RIPAS can also be enabled to broadcast alerts and warnings to pilots and ATCOs, which results in a reduction in the reaction time.

Figure A.3: Left: An Integrated Moving Map (IMM) used in the European Airport Movement Management by A-SMGCS 2 (EMMA2) project. Right: Electronic Moving Map (EMM) used during a NASA field and simulation test. In both maps, the runways have a lighter color than the taxiways. The EMM on the right shows the route of the aircraft. These images are reprinted from NASA and EMMA studies [109, 56].

ATCO HMI

ATCOs use HMI in a different way than pilots. For surveillance functions, the display function of the HMI is required. For control and guidance, ATCOs must input information into the system, e.g., for route deviation detection, a RIPAS must have information about assigned routes to compare against surveillance information.

- Electronic Flight Strips (EFS): EFSs can be used to detect conflicting clearances or to predict bottleneck situations in runway usage [32].

- Interactive Airport Maps (IAM): IAM are GUIs for ATCOs that show the airport and the traffic situation as assessed by the surveillance equipment of the airport. In addition to displaying alerts and conflicts, IAM could support automatic routing functions and could integrate information from other sources such as electronic flight strips. They operate best with cooperative targets, where reliable information on

speed and heading are available and are shown at the display next to the vehicle or the aircraft.

- Audible Alert: ATCOs often do not want to look away from the traffic during work [104] Therefore, they could miss warnings or alerts on a display. Audible alerts draw the attention of ATCOs to dangerous situations.
- Tactile switches provide haptic feedback that enables ATCOs to work without checking a display for the confirmation of triggered signals [86].

Figure A.4: Air Traffic Controller (ATCO)-Human Machine Interfaces (HMI): Left: Airport Surface Detection Equipment- Model X (ASDE-X) ATCO display. Right: Advanced Surface Movement Guidance and Control System (A-SMGCS) display. Images are reprinted from NASA and EMMA reports [110, 79].

A.4 Airport Traffic Signals

When used together and with an automatic guidance system, the following signals provide a reasonable safety net that is independent of cockpit based systems. Field studies of the European Organisation for the Safety of Air Navigation (EUROCONTROL) and the FAA found that these signals, even when used as partial solutions, had a large impact on the runway safety and reduced the number and severity of runway incursions significantly. Figure A.4 shows a schematic of the Runway Status Lights (RWSL) that uses some of the signals that are introduced here.

Runway Guard Lights (RGL) Runway guard lights (RGL) are lights that illuminate hold lines (stop bars). The International Civil Aviation Organization (ICAO) recommends one red light every 3 m on the hold line. The lights should be visible under all weather conditions. A major deficit is that the usage of stop bars differs across the major airports in Europe, leading to an inability of flight crews to deal with a red stop bar. There appears to be evidence that, at some airports, pilots are advised to cross illuminated stop bars as a matter of routine, leading to confusion among pilots.

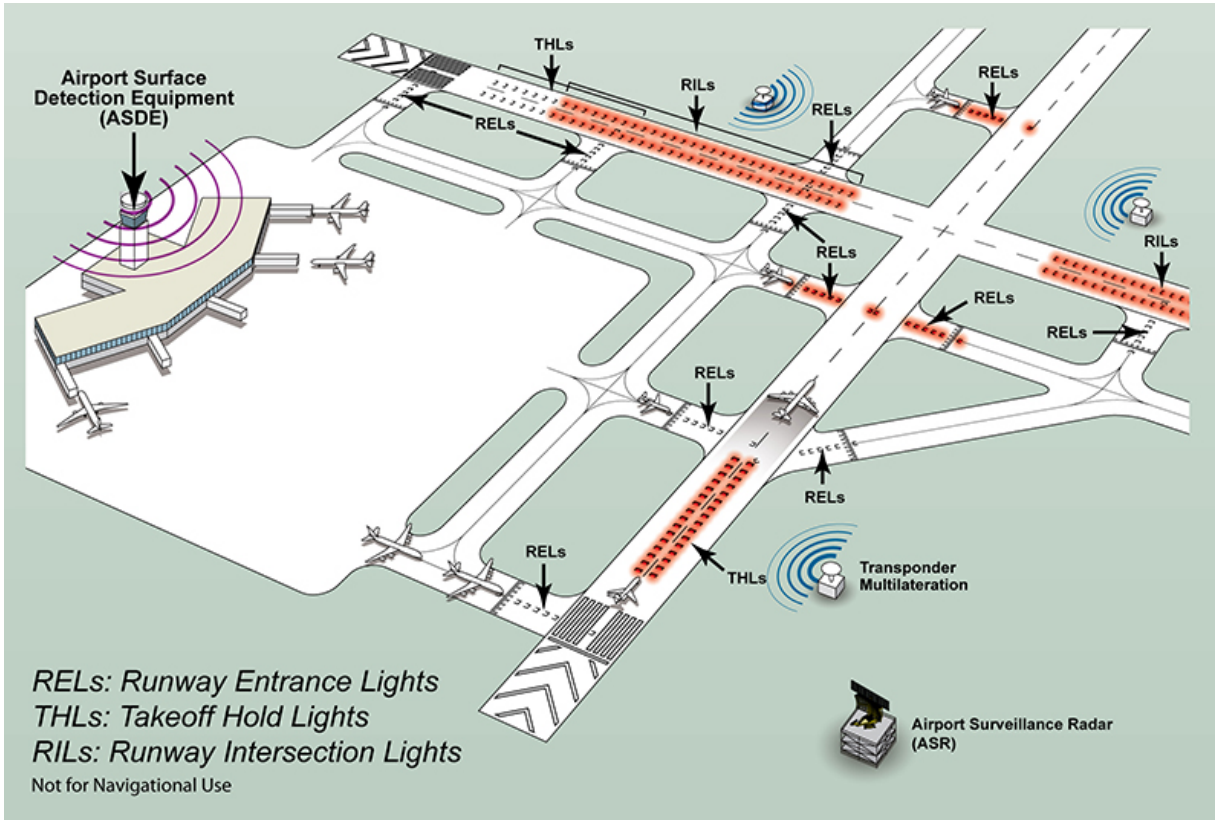


Figure A.5: Runway Entrance Lights (REL), Taxiway Hold Lights (THL), and Runway Intersection Lights (RIL) with input from Airport Surface Detection Equipment (ASDE) compose a safety net. Graphic courtesy MIT Lincoln Laboratory and FAA [65].

RGLs provide a better situational awareness of flight crews if they are consistent with RTS. RGLs were found to be ignored by flight crews under certain conditions in a study by EUROCONTROL. The same study found that RGLs are more visible if LED lights are used instead of tungsten bulbs and recommends that the distance between two adjacent lights should be 1.5 m instead of the 3 m recommended by the ICAO. One reason is that the 3 m stop bar configuration has been found to be not immediately distinguishable from taxiway and other background lights [86].
The manual setting of stop bars by the ATC does not increase the workload excessively and directs the attention of the ATCOs to the hold line operations. However, during the EUROCONTROL study mentioned above, it was found that ATCOs sometimes forget to turn on the stop bars when a vehicle has passed the hold line. It is, therefore, recommended to take an approach similar to the Hamburg Airport, where stop bars are turned on automatically as soon as a vehicle has passed the stop bar.
RGLs alone lack any means of directly signaling a stop bar violation to pilots because the pilots cannot see the RGLs once they have violated the stop bar. The left image in Figure A.4 shows the positioning of the RGLs and the way that they work.

Runway Entrance Lights Runway Entrance Lights (RELs) are red lights that are positioned at runway entrances from the hold line along the centerline of the taxiway to

Figure A.6: Left: Runway Guard Lights (RGLs), as proposed by the FAA. Right: Runway Entrance Lights (RELs), as proposed by the FAA. Graphic courtesy MIT Lincoln Laboratory and FAA [65].

the centerline of the runway. They can be illuminated in case of a stop bar violation or in case of high speed traffic on the runway to notify vehicles approaching the runway from the taxiway of the dangerous situation and the runway activity status. RELs are available as LEDs that are embedded in the pavement.
RELs can also be used to notify a vehicle if it caused a stop bar violation. However, RELs are only visible for a short time for the pilot/driver if the vehicle is moving fast. Therefore, to use RELs in such a way, the delay between the violation of the stop bar and the activation of the RELs must be minimal. Some systems attempt to predict whether a vehicle is about to overrun a stop bar; however, in a simulation experiment, it was found that the general surveillance requirements, as specified for ASDE-X and A-SMGCS, do not permit a reliable prediction unless the speed of the vehicle is very high and the distance to the stop bar is very close. A solution that has been found to work in our simulation is the use of automotive sensors for the surveillance of hotspots. These sensors can measure the speed of a vehicle using a Doppler effect with update rates of up to 30 Hz. Alternative technologies for predicting a stop bar violation are microwave fences and inductive loops. To reliably determine the speed of a vehicle, multiple fences or loops are necessary. The right image in Figure A.4 shows the positioning of the RELs and the way that they work.

Lead On Lights In contrast to RELs, Lead On Lights (LOL) show that it is safe to enter the runway from a taxiway. They are positioned similar to REL installations but are green. They create a better reassurance of ATC clearance than just turning off the illumination of a stop bar. Moreover, the LOL are still visible when the RGLs are no longer visible because the aircraft is too close to the stop bar. Intelligent multicolor LED lights embedded in the pavement are ideal for the combined installation of RELs and LOLs. Figure A.4 gives an impression of what an LOL installation could look like.

Runway Intersection Lights (RIL) Runway intersection lights are lights that are embedded in the pavement of the runway in the vicinity of runway crossings. These lights are turned on in case of a conflicting usage of the runways, e.g., two aircraft starting to move along crossing runways. The lights need to cover a long distance to provide signals at a reasonable distance to the runway crossing. The left image in Figure A.4 shows the positioning of the RILs and the way that they work.

Taxiway Hold Lights (THL) Taxiway hold lights signal to planes that are ready to begin their take off roll that the runway is not safe because of interfering traffic from a

Figure A.7: Right: Simulated Lead On Lights (LOL), as seen by the virtual pilot of a Lear jet. Left: External view of the same situation as in the right image, showing the aircraft positioned just in front of the hold line. Both images are screenshots from the Microsoft Flight Simulator (FSX) Software; the LOLs were implemented using the FSX-SDK.

taxiway, e.g., in case of a stop bar violation. These lights indicate that pilots should not take off but instead should wait until the lights turn off. The right image in Figure A.4 shows the positioning of the THLs and the way that they work.

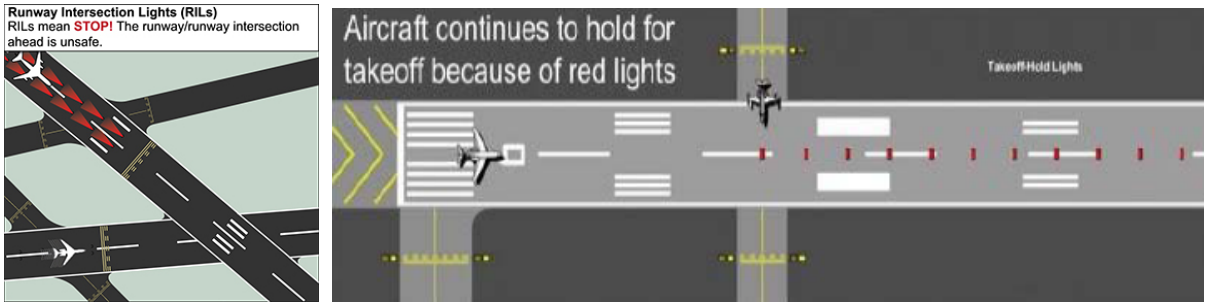


Figure A.8: Runway Intersection Lights (RIL) and Taxiway Hold Lights (THL), as proposed by the FAA. Graphic courtesy MIT Lincoln Laboratory and FAA [65].

Final Approach Runway Occupancy Signal Part of Final Approach Runway Occupancy Signal (FAROS) sets the Precision Approach Path Indicator (PAPI) lights to flashing mode, indicating to aircraft that the runway is occupied during the final approach. An FAA study found that the flashing of the PAPI did not result in a significant decrease in the final approach route of the landing aircraft [44].

Follow the Green Follow the green is an approach for avoiding route deviations. Green lights are positioned along the centerline of the runways and taxiways to lead vehicles from one place in the airport to another. The lights are turned on and off in such a way that they form a pattern moving in the direction of the route at selectable speeds.

Appendix B

Unscented Transform

The unscented transform is a transformation developed to deal with non linear transformations of linear defined probability density functions. The underlying idea is that is is easier to approximate the transformed probability density than to approximate the transformation. The unscented transform has been developed by Julier and Uhlmann in 1997, see [61, 60] a brief introduction can be found in [111]. The unscented transform aim to capture and preserve the moment of the probability density function.
For most sensor data given in polar coordinates the transformation is straightforward, the σ-points, for each dimension $\mu - \sigma, \mu, \mu + \sigma$, are transformed from polar coordinates to Cartesian coordinates and the mean μ and covariance Σ is then calculated from the transformed sigma points. The error induced by this transformation is strongly dependent from the amount of uncertainty of the sensor measurement and the distance to the sensor, see B.2

Generation of the sigma points

For the selection of the sigma points X three parameters have significant influence:

- n: The number of dimensions of $N(\mu, \Sigma)$. For each dimension at least two sigma point have to be chosen to capture the principal moments of the distribution in each dimension.
- α: A scaling parameter that determines together with κ how far the sigma points are spread from the mean.
- κ: A scaling parameter that determines together with α how far the sigma points are spread from the mean.
- β: Another parameter that can be used to encode higher order knowledge about the distribution, for Gaussian distributions 2 is optimal [111].
- $\lambda = a^2(n + \kappa) - n$: Intermediate parameter that describes the spread of the sigma points.

In the algorithm the following terms are of special interest:

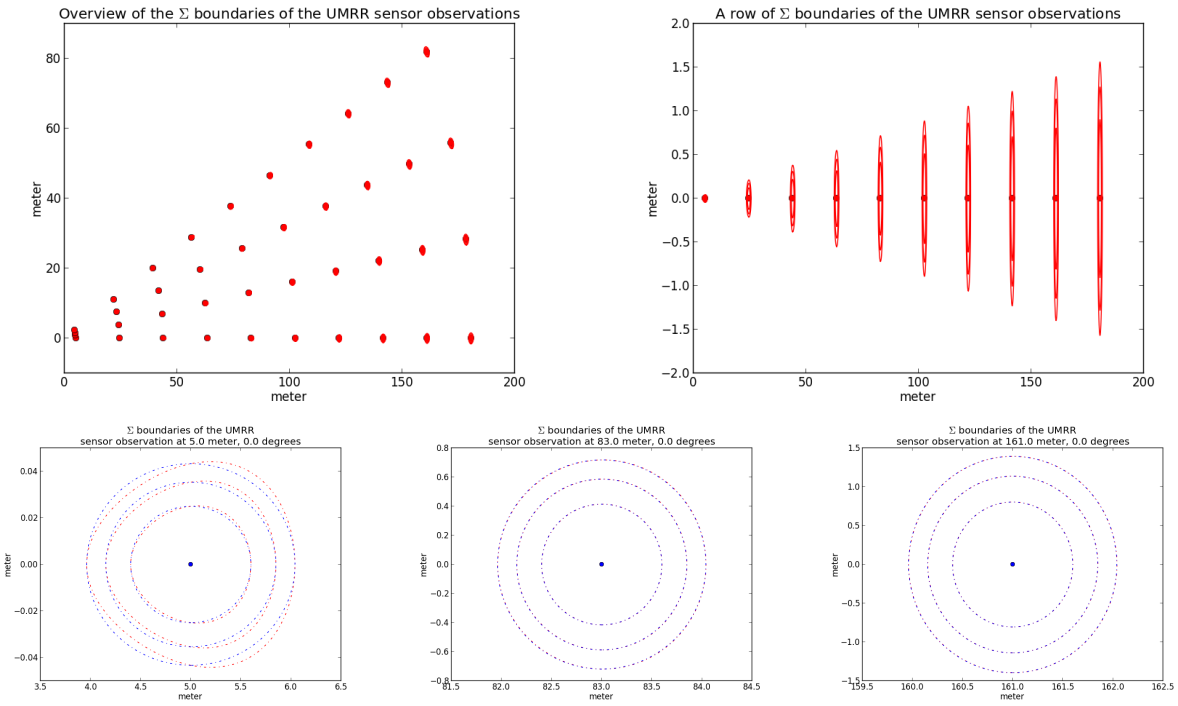


Figure B.1: The unscented transform applied to medium quality SMR measurements. The top row shows the sigma boundaries mapped to Cartesian space for distances and angles common to airport traffic surveillance. The lower row shows the detailed sigma boundaries of close, medium, and long distance measurements and the sigma boundaries of the approximation by the unscented transform. The original probability density function is plotted in red.

- $\left(\sqrt{(n+\lambda)\Sigma}\right)_i$: The ith column of the lower triangular of the Cholesky decomposition L of $(n+k)\Sigma$. [60]. For the trivial case where the dimensions are independent, meaning that only the diagonal entries contain $\sigma_i^2, i \in n$ L is a matrix with σ_i as diagonal entries. For other cases where the entries outside the diagonal are not 0 the matrix is lower triangular with $(n+k)\Sigma = LL^T$ which looks like e.g.:

$$\left(\sqrt{(n+\lambda)\Sigma}\right)_i = \begin{pmatrix} 41.55997709 & -0.0 \\ -66.48258378 & 22.31036224 \end{pmatrix} \tag{B.1}$$

$$X^0 = \mu \tag{B.2}$$

$$X^i = \mu + \left(\sqrt{(n+\lambda)\Sigma}\right)_i, i \in 1, ..., n \tag{B.3}$$

$$X^i = \mu - \left(\sqrt{(n+\lambda)\Sigma}\right)_{n-i}, i \in n+1, ..., 2n \tag{B.4}$$

Generation of weights

Attached to each sigma point X_i is a weight w_i. The weight w_0 is a special case as it is different for computation of the covariance Σ and the computation of the mean μ.

$$w_0^{\Sigma} = \frac{\lambda}{(n+\lambda)} * (1.0 - a^2 + \beta) \tag{B.5}$$

$$w_0^{\mu} = \frac{\lambda}{n+\lambda} \tag{B.6}$$

$$w_i = \frac{1}{2(n+\lambda)}, i \in 1, ..., n \tag{B.7}$$

$$w_i = \frac{1}{2(n+\lambda)}, i \in n+1, ..., 2n \tag{B.8}$$

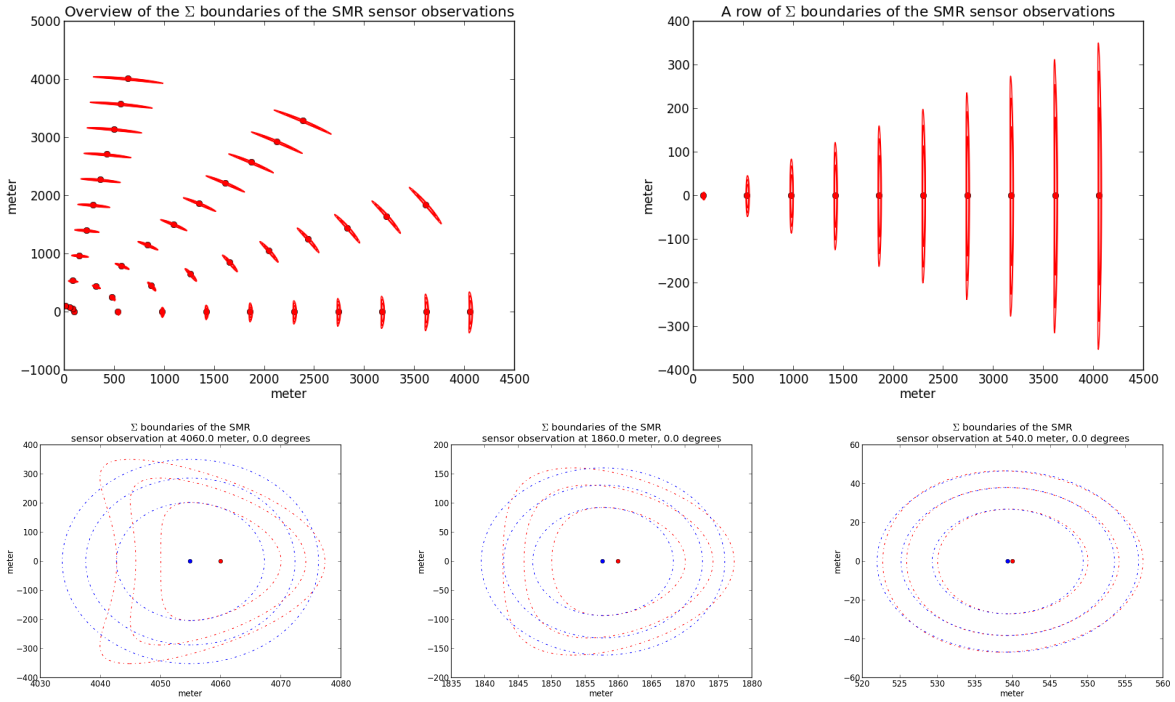


Figure B.2: The unscented transform applied to medium quality SMR measurements. The top row shows the sigma boundaries mapped to Cartesian space for distances and angles common to airport traffic surveillance. The lower row shows the detailed sigma boundaries of close, medium, and long distance measurements and the sigma boundaries of the approximation by the unscented transform.

Computation to the new mean and covariance

The new mean μ^{new} and new covariance Σ^{new} are computed using the weighted transformed sigma points $f(X_i)$. The new mean is computed using the equation:

$$\mu^{new} = w_0^{\mu} \sum_{i=1}^{2n} w_i f(X_i) \tag{B.9}$$

The new covariance is computed using the equation:

$$\Sigma^{new} = w_0^{\Sigma} \sum_{i=1}^{2n} w_i \left(X_i \times X_i^T \right) \tag{B.10}$$

The unscented transform applied to the polar to Cartesian problem

When applying the unscented transform to the problem of converting normal distributions from polar coordinates to Cartesian coordinates the results strongly depend on the parametrization of the normal distribution. The bigger the uncertainty, the bigger the error of the approximation. The transformation is quite precise for UMRR measurements and Laser measurements, but has a significant error for long distance measurements of older SMR.
Figure B.2 and Figure B.1 provide an overview of the sigma boundaries for SMR measurements and UMMR measurements of objects in a reasonable distance and angle to the sensor. Also, the figures show detailed plots of example measurements that display the nature of the error induced by the unscented transform. The leftmost plot in Figure B.2 shows that the transformed probability density function covers an area that is not covered by the original probability density function.

Appendix C

Formal Complexity of the Data Association Problem

This appendix covers the complexity of the data association problem. The first section states the formal complexity analysis based on publications on the topic, the second section aims to explain the cause for the complexity by example, and the third section gives a brief overview of approaches to deal with the complexity in real time applications.

C.1 Formal Complexity in the Literature

The established definition of data association is :"'The process of assign and compute the weights that relates the observations of tracks (A track can be defined as an ordered set of points that follow a path and are generated by the same target.) from one set to the observation of tracks of another set."' [42].
The formally correct treatment of the data association problem is considered to be NP-Hard[1]. May N be the number of observations and tracks for a series of discrete times. A proper treatment of the data association problem requires the proper probabilistic treatment of all combinations of possible associations between tracks and observations. This requirement raises the number of possible assignments to $N!$ [17] , see Figure C.1 for an example. The computation is considered equivalent to the problem of computing the permanent of the matrix [17] that has been shown to be NP-Hard[114], which implies that the data association problem is also NP-Hard [17]. While even solving the problem only for the time t_0, t_1 is NP-Hard, the problem gets increasingly complex as time passes. Consider a series of k discrete times. For the proper probabilistic treatment the number of possible mappings is $N!^{k-1}$, were only one mapping is the true mapping from observations to objects/tracks. A detailed explanation and proof can be found in[17] and [114]. For many real world scenarios effects such as missing observations, false positives, multiple observations per object, and multiple objects causing a single observation significantly

[1]According to [46] NP is the class of problems that can be solved by a non deterministic Turing-machine in polynomial-time. NP-hard (Non deterministic Polynomial-time hard), in computational complexity theory, is a class of problems for which is at least as hard as any problem in NP but maybe even harder, so it is not NP-complete, which would require the problem to be part of NP. Such problems are considered intractable, as it requires to much time to compute their solution, except for special small instances.

increase the complexity of the data association.

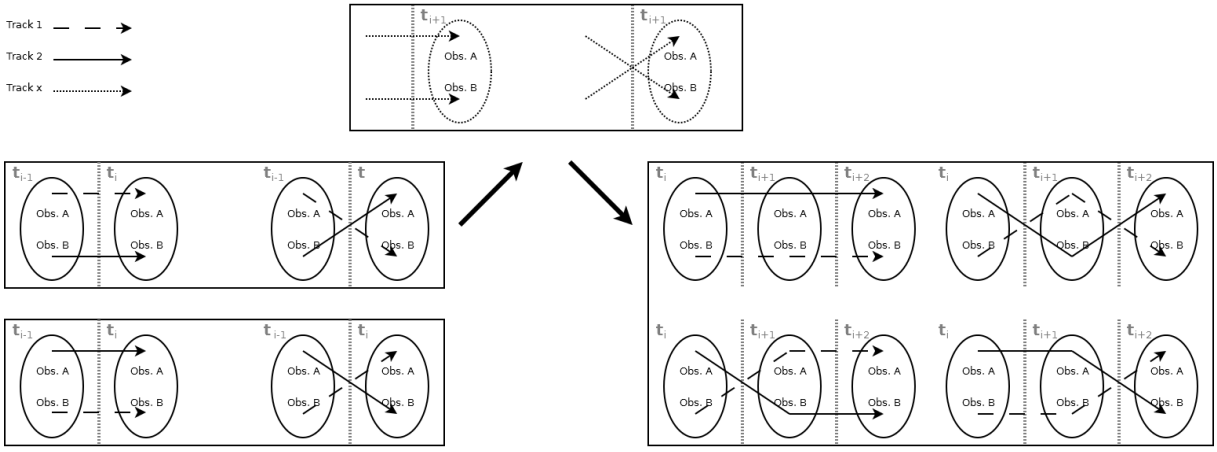


Figure C.1: The 2 leftmost boxes show the possible data associations for 2 tracks and 2 observations ($N = 2$) for 2 sets of observations of tracks from times $ti - 1$ and t_i ($k = 2$). There are four possibilities to assign the tracks to the observations, but from these two pairs are semantically equal, so only $N!^{k-1} = 2^1 = 2$ combinations remain. Each additional consecutive set of observation that is added to the series could result in the data association shown in the top box and effectively multiplies the number of possible mappings by $N! = 2$ resulting in $N!^{k-1} = 2^2 = 4$ possible data associations for a series of 3 ($k = 3$) time steps, shown in the rightmost box.

C.2 Explanation by Example.

Consider the data association of sensor readings at a single time step t_0 where the number of objects N is known[2], and there are N detected features in the sensor readings that might be caused by any object. These features will be referred to as observations in the following text. Each object may be the cause for any observation(s) or for no observation. Each observation may also be caused by any object(s) or no object at all and relate to an error of a sensor. Displayed as a so called data association matrix this looks like the table shown in Figure C.2.
So the number of possible data association matrices, using only 1 and 0 weights, is 2^{NN}. Here a cell with a 1 indicates a causal relationship from the row of the cell to the column of the cell, meaning that the row's object caused the column's observation. A 0 indicates that there is no causal relation between the according object and the observation. The rightmost column has a special role as a 1 indicates that the object did not cause any observation in the timestep represented by the matrix. A 1 in the bottom row indicates that the observation is a false alarm and was not caused by any of the tracked objects but by some other factor. In the consequence a 1 in the lower right cell indicates that there is no missing observation, and no false alarm.
This includes some impossible configurations, which have a probability of 0. These impossible combinations are all matrices where there is a zero row or a row with a 1 in the

[2]Often the number of objects is unknown, it will be shown that the problem is NP-Hard even with a known number of objects.

Data Association Matrix t0

	Observation 1	Observation ...	Observation M	No Observation
Object 1				
Object ...				
Object N				
No Object				

Figure C.2: A data association matrix, as explained in the text.

last column and at least another 1. Also impossible are columns with a 1 in the bottom row and at least another 1 in any row.
E.g.: The m data association matrix for two objects and two observations has 3 rows: The first row for object 1, the second row for object 2, and the third row for false alarms. The matrix has 3 columns: The first column for the first observation, the second column for the second observation, and the third column for the missing observation.
While even solving the problem only for the time t_0, t_1 is NP-Hard, the problem gets increasingly complex as time passes. Consider the case that there are ten data association matrices for k consecutive times $(t_1 \dots t_k)$ where the number of objects and observations remains constant. For an exact treatment of the data association problem one has to consider not only all data association matrices for each time, but rather has to consider all possible sequences of data association matrices of all consecutive times. The number of possible sequences is then $(2^{NN})^k$. For the proper probabilistic treatment the number of possible sets is $N!^{k-1}$, were only one set is the true mapping from observations to objects/tracks.

C.3 Approaches for Data Association in Real Time Applications

Fortunately algorithms and methods for the computation of approximations of the data association exist that are much faster and yield reasonable good results. These approaches reduce the complexity of the computation based on assumptions about the causality.
A significant reduction of the complexity can be achieved by the assumption that, given the data association at the discrete time t is correct and the formal representation of tracks holds all relevant information for the correct data association, the data association can be done based on the information available at discrete times t_{i-1} and t_i. Since the mathematical models and the formal representation are abstractions and do not hold all relevant information for all situations algorithms based on this assumption do work very well in controlled environments as long as the objects do behave as expected.

Another way to significantly reduce the complexity of the data association is to ignore associations between tracks and observations that are so unlikely that they are considered to be negligible, and use algorithms and data structures that reduce the number of possible associations by systematically including only the associations that have a significant probability. The number of different implementations of this approach depend on many factors and are so numerous that an introduction is beyond the scope of this work. Therefore the following sketch of a method based on this approach should serve to explain the underlying idea in brief.
Consider that observations and tracks are represented by the same set of N variables. The data structure that holds the information on the observation and tracks is a grid of overlapping cells over the vector space spanned by the variables. Observations and tracks are mapped to the cells of the grid using the most significant part of the variables and the association between an observation and an object is only computed for objects and observations in the same cell of the grid. Given that the separation distance between tracks is usually shorter than the diameter of a cell and the length of the error vector of the observations is also shorter than the diameter of a cell, this approach approximates the exact probabilistic computation of the data association, in the best case with a complexity of $O(M)$ for M Tracks and M Observations at a given time t, by ignoring unlikely associations between tracks and observations.
If both approaches are combined it is - under very favorable circumstances - possible to reduce the complexity of the data association to $O(min(N, M) * k)$ for N Observations, M tracks and k discrete times.

Acknowledgments

If it wasn't for the aid and support of many people i wouldn't have completed this work by now.
I would like to express my gratitude to my supervising tutor Prof. Dr.-Ing. D.P.F. Möller, for encouraging me to take this challenge, helping me to enter the scientific community, and giving me the freedom to pursue my scientific ideas.
Moreover i would like to thank Prof. Dr. F. Mayer-Lindenberg for the inspiring discussions on the technical details of my work that helped me to sharpen up my understanding of these topics and refine my work.
My colleague Steven Köhler was always open to discuss almost any topic and this helped me a lot to reflect my ideas. His expertise in software engineering and the C++ programming language surely saved me a tremendous amount of time.
The members of the TIS Research Group , namely Carola Tenge, Dr. Kai Himstedt, Dieter Florstedt, Michael Borchers, and Dr. Werner Hansman deserve my gratitude for making my time at the TIS Research Group a good one and helping me out on many occasions.
Above all, I would like to thank my wife Sirit an my sons Thore, Isbjörn, and Tristan for all the support and understanding during the completion of this work. I cannot adequately acknowledge the ways in which they have contributed to my work.
Sincere thanks go to the German tax payers, as this work would not have been possible without the financial support of the Federal Ministry of Economy and Technology (BMWi) for parts of this work have been funded by in the national research program "Competitive Airport" (WFF) by the BMWI's special program LuFo IV.
Thanks go also to Detlef Schnathmann and Thomas Kamm from our project partner ADB / Siemens Airfield Solutions. Their expertise, visions, and the inspiring discussions helped me to find a place for my research in the jungle of the airfield stakeholder interests.

List of Figures

List of Tables

Bibliography

[1] AATL. Dsmr-800 surface movement radar. Technical report, AATL, 2011.

[2] Anonymous. Surface movement radar system performance capabilities, [online]. available: http://www.scribd.com/doc/44616594/smrperformance, October 2010.

[3] Federal Aviation Administration (FAA) ATO-Terminal Services, Surveillance Sector. Performance specification runway status lights (rwls) system, March 2008.

[4] Yaakov baar Shalom, Fred Daum, and Jim Huang. The probabilistic data association filter. *IEEE Controls Systems Magazine*, December 2009.

[5] Marie-Luise Berry. Summary report - development and validation of improvement of runway safety net (a-smgcs level 2) by eletronic flight progress strips - contract t07/1160hg. Technical report, EUROCONTROL, 2010.

[6] Albrecht Beutelspacher. *Lineare Algebra*. vieweg, February 2001.

[7] M. Bolic, P.M. Djuric, and Sangjin Hong. Resampling algorithms and architectures for distributed particle filters. *Signal Processing, IEEE Transactions on*, 53(7):2442–2450, July 2005.

[8] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*. USENIX Association, 2010.

[9] Rick Cassel. Development of the runway incursion advisory and alerting system (riaas). Technical report, NASA, 2005.

[10] Rick Cassel, Carl Evers, Jeff Esche, and Benjamin Sleep. Nasa runway incursion prevention system (rips) dallas-fort worth demonstration preformance analysis. Technical report, NASA, 2002.

[11] R. Cassell, C. Evers, B. Sleep, and J. Esche. Initial test results of pathprox - a runway incursion alerting system. In *NASA*, volume 1, 2001.

[12] Caroline Chabrol. Cdg a-smgcs test report. Technical report, EUROCONTROL, 2010.

[13] Joseph R. Chambers. Innovation in flight: Research of the nasa langley research center on revolutionary advanced concepts for aeronautics. Technical report, NASA, Hampton, Virginia, USA, 2005.

[14] Julie Chambon-Dupont-Ferrier. Rapport. Master's thesis, 2008.

[15] Robert Chapin. Runway incursions 2000-2010: Is safety improving?, 2010.

[16] Munawar Chaudhary, David Hilton, and Richard Allen. Runway safety at london heathrow airport, 2008.

[17] J.B. Collins and J.K. Uhlmann. Efficient gating in data association with multivariate gaussian distributed states. *Aerospace and Electronic Systems, IEEE Transactions on*, 28(3):909–916, Jul 1992.

[18] Sensis Cooperation. *Airport Surface Detection Equipment, Model X*, January 2011.

[19] Dan Crisan and Arnaud Doucet. A survey of convergence results on particle filtering methods for practitioners, 2002.

[20] L. Cristaldi, G. D'Antona, M. Faifer, A. Ferrero, and R. Ottoboni. Aircraft classification using a microwave barrier. In *Measurement Systems for Homeland Security, Contraband Detection and Personal Safety, Proceedings of the 2006 IEEE International Workshop on*, pages 44 –48, oct 2006.

[21] F. Daum and J. Huang. Curse of dimensionality and particle filters. In *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, volume 4, pages 1979–1993, 8-15, 2003.

[22] Florent de Dinechin, Jérémie Detrey, Octavian Cret, and Radu Tudoran. When fpgas are better at floating-point than microprocessors. In *Proceedings of the 16th international ACM/SIGDA symposium on Field programmable gate arrays*, FPGA '08, pages 260–260, New York, NY, USA, 2008. ACM.

[23] K. Dimitropoulos, N. Grammalidis, I. Gragopoulos, H. Gao, Th. Heuer, M. Weinmann, S. Voit, C. Stockhammer, U. Hartmann, and N. Pavlidou. Detection, tracking and classification of vehicles and aircraft based on magnetic sensing technology. *World Academy of Science, Engineering and Technology*, 2006.

[24] Arnaud Doucet, Nando De Freitas, Neil Gordon, et al. *Sequential Monte Carlo methods in practice*, volume 1. Springer New York, 2001.

[25] Arnaud Doucet, Nando de Freitas, Kevin P. Murphy, and Stuart J. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 176–183, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[26] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *STATISTICS AND COMPUTING*, 10(3):197–208, 2000.

[27] Bruce Powel Douglass. *Real-Time Agility: The Harmony/ESW Method for Real-Time and Embedded Systems Development.* Addison-Wesley Professional, 2009.

[28] V. Edwards, C. Evers, and E. Chartier. Inductive loop sensor subsystem (lss) as a supplemental surface surveillance system-demonstration results. In *Digital Avionics Systems Conference, 2000. Proceedings. DASC. The 19th*, volume 2, pages 7D6/1 –7D6/8 vol.2, 2000.

[29] EUROCAE. Minimum operational performance specification for mode s multilateration systems for mode s multilateration systems for use in a-smgcs, ed-117. Technical report, EUROCAE, 2003.

[30] EUROCONTROL. Annual safety report 2009. Technical report, EUROCONTROL, 2009.

[31] EUROCONTROL. Anual safety report 2010. Technical report, EUROCONTROL, 2010.

[32] EUROCONTROL. Development and validation of improvment of runway safety net (a-smgcs level 2) by electronic flight strips - d16 cost benefit analysis. Technical report, EUROCONTROL, 2010.

[33] Federal Aviation Administration (FAA). Fact sheet faa adopts icao definition for runway incursions, October 2007.

[34] Emily B. Fox, David S. Choi, and Alan S. Willsky. Nonparametric Bayesian methods for large scale multi-target tracking. In *Proc. Asilomar Conf. Signals, Systems, and Computers*. IEEE, November 2006.

[35] Emily B. Fox, Erik B. Sudderth, and Alan S. Willsky. Hierarchical Dirichlet processes for tracking maneuvering targets. In *Proc. International Conference on Information Fusion*. IEEE, July 2007.

[36] S.J. Godsill, J. Vermaak, K-F. Ng, and J-F. Li. Models and algorithms for tracking of manoeuvring objects using variable rate particle filters. *Proc. IEEE*, 95(5), May 2007.

[37] G. Govindu, L. Zhuo, S. Choi, P. Gundala, and V. Prasanna. Area and power performance analysis of a floating-point based application on FPGAs. 2003.

[38] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *Signal Processing, IEEE Transactions on*, 50(2):425–437, Feb 2002.

[39] Fredrik Gustafsson, Thomas B. Schï¿½n, Rickard Karlsson, and Per-Johan Nordlund. State-of-the-art for the marginalized particle filter. In *Nonlinear Statistical Signal Processing Workshop*, September 2006.

[40] David Lee Hall. *Mathematical Techniques in Multisensor Data Fusion*. Artech House, Inc., Norwood, MA, USA, 1992.

[41] D.L. Hall and J. Linas. *Handbook of Multisensor Data Fusion*. CRC Press, May 2001.

[42] D.L. Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, Jan 1997.

[43] D. Held, J. Levinson, and S. Thrun. A probabilistic framework for car detection in images using context and scale. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1628 –1634, may 2012.

[44] John Helleberg. Effects of a final approach runway occupancy signal (faros) on pilots' flight path tracking, traffic detection, and air traffic control communications. Technical report, The Mitre Cooperation, 2005.

[45] Sangjin Hong, M. Bolic, and P.M. Djuric. An efficient fixed-point implementation of residual resampling scheme for high-speed particle filters. *Signal Processing Letters, IEEE*, 11(5):482 – 485, May 2004.

[46] John E. Hopcroft and Jeffrey D. Ullman. *Einfï¿½hrung in die Automatentheorie, Formale Sprachen und Komplexitï¿½tstheorie*. Oldenbourg R. Verlag GmbH, 2002.

[47] C. Hue, J. Le Cadre, and P. Perez. Tracking multiple objects with particle filtering, 2000.

[48] ICAO. *Doc 9830 - AN/452, Advanced Surface Movement and Control Systems (A-SMGCS) Manual*, 2004.

[49] ICAO. *Doc 9870 - AN/463, Manual on the Prevention of Runway Incursons*. ICAO, 2007.

[50] Deutsches Zentrum für Luft-und Raumfahrt Institut für Flugführung. A-smgcs experimental system in braunschweig, 2008.

[51] Intel. Ark - your source for intel product infomation, Sep 2014.

[52] Air Line Pilots Association International. White paper - runway incursions - a call for action, 2007.

[53] J. Jakobi. Emma2 recomendations report. Technical report, DLR, 2009.

[54] Denise .R. Jones. Runway incursion prevention system simulation evaluation. In *Digital Avionics Systems Conference, 2002. Proceedings. The 21st*, volume 2, pages 11B4–1– 11B4–12vol.2, 2002.

[55] Denise R. Jones and Lawrence J. Prinzel. Runway incursion prevention for general aviation operations. In *25th Digital Avionics Systems Conference, 2006 IEEE/AIAA*, pages 1–12, Hampton, VA, 2006. NASA.

[56] Denise R. Jones and Lawrence J. Prinzel. Simulator evaluation of runway incursion prevention technology for general aviation operations. Technical report, NASA, Hampton, Virginia, USA, 2011.

[57] D.R. Jones, C.C. Quach, and S.D. Young. Runway incursion prevention system-demonstration and testing at the dallas/fort worth international airport. In *Digital Avionics Systems, 2001. DASC. 20th Conference*, volume 1, pages 2D2/1–2D2/11 vol.1, 2001.

[58] Tammy Jones. Fact sheet - airport surface detection equipment, model x (asde-x), October 2010.

[59] David F. Green Jr. Runway safety monitor algorithm for runway incursion detection and alerting. Technical report, NASA, 2002.

[60] Simon J. Julier, Jeffrey Uhlman, and Hugh F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3), March 2000.

[61] Simon J. Julier and Jeffrey K. Uhlmann. Reduced sigma point filters for the propagation of means and covariances through nonlinear transformations. In *In Proceedings of the 2002 American Control Conference*, pages 887–892, 2002.

[62] K. Klein and J. Jakobi. Beta recommendations report - operational benefit evaluation by testing an a-smgcs. Technical report, DLR, 2003.

[63] J. Ko, D.J. Klein, D. Fox, and D. Hahnel. Gp-ukf: Unscented kalman filters with gaussian process prediction and observation models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

[64] Cody C. T. Kwok and Dieter Fox. Map-based multiple model tracking of a moving object. In Daniele Nardi, Martin Riedmiller, Claude Sammut, and José Santos-Victor, editors, *RobuCup*, volume 3276 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 2004.

[65] MIT Lincoln Laboratory. Runway incursion status lights, December 2010.

[66] E. Le Grand and S. Thrun. 3-axis magnetic field mapping and fusion for indoor localization. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, pages 358 –364, sept. 2012.

[67] Nancy G. Leveson. *Safeware: system safety and computers.* ACM, New York, NY, USA, 1995.

[68] X. R. Li and V. P. Jilkov. A survey of maneuvering target tracking-part ii: Ballistic target models. In *Proc. 2001 SPIE Conf. Signal and Data Processing of Small Targets*, volume 4473, pages 559–581, July-August 2001.

[69] X. R. Li and V. P. Jilkov. A survey of maneuvering target tracking-part iii: Measurement models. In *Proc. 2001 SPIE Conf. Signal and Data Processing of Small Targets*, volume 4473, pages 423–446, July-August 2001.

[70] X. R. Li and V. P. Jilkov. A survey of maneuvering target tracking-part iv: decision-based methods. In *Proc. 2002 SPIE Conf. Signal and Data Processing of Small Targets*, volume 4728-60, April 2002.

[71] X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking-part i: dynamic models. *IEEE Transactions on Aerospace and Electronic Systems*, 4(39), October 2003.

[72] X. R. Li and V. P. Jilkov. A survey of maneuvering target tracking: approximation techniques for nonlinear filtering. In *Proc. 2004 SPIE Conf. Signal and Data Processing of Small Targets*, volume 5428-62, pages 537–550, April 2004.

[73] X. R. Li and V. P. Jilkov. Survey of maneuvering target tracking. part v: multiple-model methods. *IEEE Transactions on Aerospace and Electronic Systems*, 4(41), October 2005.

[74] Zhe Liu, Weidong Chen, Yong Wang, and Jingchuan Wang. Localizability estimation for mobile robots based on probabilistic grid map and its applications to localization. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, pages 46–51, 2012.

[75] Christoph Meier. *Datenfusionsverfahren für die automatische Erfassung des Rollverkehrs auf Flughäfen*. PhD thesis, Fakultät für Maschinenbau und Elektrotechnik der Technischen Universität Carolo-Wilhelmina zu Braunschweig, 1998.

[76] H.B. Mitchell. *Multi-Sensor Data Fusion, An Introduction*. Springer, 2007.

[77] H. Miyazaki, H. Koga, E. Kakubari, and S. Nihei. Evaluation of multilateration at narita international airport. Technical report, Electronic Navigation Research Institute, Japan, 2010.

[78] H. Miyazaki, T.K. Ueda, E. Kakubari, and S. Nihei. Evaluation results of airport surface multilateration. In *EIWAC2010*, 2010.

[79] NASA. Asde-x-display. Online, 2008.

[80] Aero News Network. Aero news network, sep 2011.

[81] National Transportation Safety Board (NTSB). Aviation incident database, report ops09ia005a, October 2009.

[82] National Transportation Safety Board (NTSB). Ntbs most wanted list - transportation safety improvements 2010 - 2011, November 2010.

[83] FAA Office of Runway Safety. Faa anual runway safety report 2007, 2007.

[84] FAA Office of Runway Safety. Faa anual runway safety report 2009, 2009.

[85] L. Ong, B. Upcroft, M.F. Ridley, T.A. Bailey, S. Sukkarieh, and H.F Durrant-Whyte. Decentralised data fusion with particles. In *Proceedings Australiasian Conference on Robotics and Automation (ACRA)*, 2005.

[86] Yvonne Page, Eric Miart, and Paul Wilson. Runway safety - use of stop bars 24h. Technical report, EUROCONTROL, 2008.

[87] Yun Pan, Ning Zheng, Qinglin Tian, Xiaolang Yan, and Ruohong Huan. Hierarchical resampling algorithm and architecture for distributed particle filters. *Journal of Signal Processing Systems*, 71(3):237–246, 2013.

[88] N. Pavlidou, N. Grammalidis, K. Dimitropoulos, D. Simitopoulos, and M.Strintzis. Video sensor data fusion for advanced surface movement guidance and control systems. In *Sixth COST 276 Workshop on Information and Knowledge Management for Integrated Media Communication*, pages 127–132, May 2004.

[89] Clemens Rabe, Thomas Müller, Andreas Wedel, and Uwe Franke. Dense, Robust, and Accurate Motion Field Estimation from Stereo Image Sequences in Real-Time. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Proceedings of the 11th European Conference on Computer Vision*, volume 6314 of *Lecture Notes in Computer Science*, pages 582–595. Springer, September 2010.

[90] J. Reuss. Investigation report ex006-1-2/04. Technical report, German Federal Bureau of Aircraft Accident Investigation, 2009.

[91] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, 2004.

[92] Thomas Röfer. Region-based segmentation with ambiguous color classes and 2-D motion compensation. In Ubbo Visser, Fernando Ribeiro, Takeshi Ohashi, and Frank Dellaert, editors, *RoboCup 2007: Robot Soccer World Cup XI*, pages 369–376. Springer-Verlag, Berlin, Heidelberg, 2008.

[93] Matthew Rosencrantz, Geoffrey Gordon, and Sebastian Thrun. Locating moving entities in indoor environments with teams of mobile robots. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 233–240, New York, NY, USA, 2003. ACM.

[94] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

[95] A.C. Sankaranarayanan, R. Chellappa, and A. Srivastava. Algorithmic and architectural design methodology for particle filters in hardware. In *Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on*, pages 275 – 280, October 2005.

[96] Simo Särkkä, Aki Vehtari, and Jouko Lampinen. Rao-Blackwellized particle filter for multiple target tracking. *Inf. Fusion*, 8(1):2–15, 2007.

[97] J. Schönefeld and D.P.F. Möller. Fast and robust detection of runway incursions using localized sensors. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, pages 33–39. IEEE, 2012.

[98] J. Schönefeld and DPF Möller. Mathematical aspects of the implementation of particle filters on fpga. In *MathMod 2012*, 2012.

[99] J. Schönefeld and DPF Möller. Runway incursion prevention systems: A review of runway incursion avoidance and alerting system approaches. *Progress in Aerospace Sciences*, 2012.

[100] Janis Schönefeld. Multi-sensor data fusion applied to airport traffic surveillance. Master's thesis, Universität Hamburg, MIN-Fakultät, Department Informatik, Arbeitsbereich TIS, April 2008.

[101] A. Seekircher, T. Laue, and T. Röfer. Entropy-based active vision for a humanoid soccer robot. In J. Ruiz del Solar E. Chown and P.G. Ploeger, editors, *RoboCup 2010: Robot Soccer World Cup XIV*, number 6556 in Lecture Notes in Artificial Intelligence, pages 1–12. Springer; Heidelberg; http://www.springer.de/, 2011. Best Paper Award.

[102] G.K. Singh and Christoph Meier. Preventing runway incursions and conflicts. *Aerospace Science and Technology*, 8:653–670, 2004 2004.

[103] Peter N. Squire, Jane H. Barrow, Kevin T. Durkee, Carl Smith, Jennifer C. Moore, and Raja Parasuraman. Rimdas: A proposed system for reducing runway incursions. *Ergonomics in Design*, 2010.

[104] Ronald K. Stevens and Julian Sanchez. Warning, runway occupied: An evaluation of tower controller behavior when maintaining runway safety. In *Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th*, pages 4.C.5–1 –4.C.5–12, October 2010.

[105] Constanze Stockhammer, Haibin Gao, Thomas Heuer, and Uwe Hartmann. Ismael - intelligent surveillance and management functions for airfield applications based on low cost magnetic field detectors, 2005.

[106] Constanze Stockhammer, Haibin Gao, Thomas Heuer, Uwe Hartmann, Kosmas-Dimitropoulos, Nikolaos Grammalidis, Ioannis Gragopoulos, Niovi Pavlidou, and Joerg Pfister. Ismael - reliable eyes for air traffic controllers at airports. In *Intelligent Transportation Systems Conference*. IEEE, 2006.

[107] Daniel Stronger and Peter Stone. A comparison of two approaches for vision and self-localization on a mobile robot. In *IEEE International Conference on Robotics and Automation*, pages 3915–3920, April 2007.

[108] Alex Teichman and Sebastian Thrun. Tracking-based semi-supervised learning. *The International Journal of Robotics Research*, 31(7):804–818, 2012.

[109] J. Teutsch. Emma2 - validation comparative analysis report. Technical report, DLR, 2009.

[110] J. Teutsch. Milan malpensa a-smgcs safety rts test report. Technical report, DLR, 2009.

[111] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005.

[112] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge: Research articles. *J. Robot. Syst.*, 23(9):661–692, 2006.

[113] Office of Inspector General U.S. Department of Transportation. Faa needs to improve asde-x management controls to address cost growth, schedule delays, and safety risks, October 2007.

[114] L.G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, pages 189–192, 1997.

[115] J. Vermaak, S. Godsill, and P. Perez. Monte carlo filtering for multi-target tracking and data association. *IEEE Tr. Aerospace and Electronic Systems*, 1(41):309–332, January 2005.

[116] Edward L. Waltz and James Llinas. *Multisensor Data Fusion.* Artech House, Inc., Norwood, MA, USA, 1990. Foreword By-Franklin E. White.

[117] Martin Weser. Multimodales tracking und trajektorien vorhersage, April 2006.

[118] Binli Ye and Yunhua Zhang. Improved fpga implementation of particle filter for radar tracking applications. In *Synthetic Aperture Radar, 2009. APSAR 2009. 2nd Asian-Pacific Conference on*, pages 943 –946, October 2009.

[119] Steven D. Young and Denise R. Jones. Runway incursion prevention: A technology solution. In *Joint Meeting of the Flight Safety Foundation's 54^{th} Annual International Air Safety Seminar, the International Federation of Airworthiness' 31^{st} International Conference, and the International Air Transport Association*, November 2001.

[120] Flughafen Zürich. Runway safety report 2008. Technical report, Flughafen Zürich, Zürich, 2008.

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorstehende Dissertation selbst verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

..... **Hamburg, Mai 2015**

Janis Schönefeld